UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Software product user feedback sentimental analysis on Twitter at cross country level

Udhaya Shankar Subramanian

August 25, 2022

Student ID: 21071225

Supervisors: Dr. Tapajit Dey

Masters of Artificial Intelligence and Machine Learning

Department of Electronic and Computer Engineering

Abstract

The popularity of social media platforms has skyrocketed during the past several years. Twitter is now the most popular microblogging service, where users exchange "tweets" (short messages) about various topics. Twitter is used for software-related communication and user feedback by a wide range of people worldwide. It includes everyone from individual engineers and software firms to end users in every corner of the globe. It has been demonstrated in the past that such comments help developers.

Totally 15072 tweets have been collected from Twitter accounts that support other social media platforms, such as Facebook, WhatsApp, Snapchat, Telegram, Instagram, and Messenger, utilizing the Tweepy API. These tweets were analyzed using the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentimental analysis approach to learn about users' feelings toward various software products. Here we are comparing the values of two groups using the Mann-Whitney U test, a rank-based test rank-based test significantly distinct from one another.

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this or any other University and that, unless otherwise stated, it is my work.

__udhaya shankar subramanian________________________

Udhaya Shankar Subramanian August 25, 2022

The dissertation work was conducted from May 2022 to August 2022 under the supervision of Dr. Tapajit Dey at the University of Limerick.

DocuSigned by:

Tapajit Dey

EB37F5BFC54B48C...

25 August 2022 | 11:35 PDT

____

Dr Tapajit Dey

ACKNOWLEDGMENTS

First and foremost, I would like to sincerely express my gratitude to Dr. Tapajit Dey for his timely guidance and support throughout this dissertation. It was a great honor to work under his supervision. I would also like to acknowledge them for their valuable suggestions, feedback, and great advice that helped me to push the limits and complete the work.

I would like to thank all my professors and lecturers who taught me different concepts of Machine Learning, which helped me throughout the course and complete the dissertation. Also, thanks to the family and friends for their constant support and encouragement. Above all, I thank the Almighty God for his blessings throughout the course.

Table of Contents

List of Figures

List of Tables

1. Introduction and Report Outline

In recent years, microblogging has exploded in popularity as a way for individuals to quickly and easily share and spread information and ideas. A common feature of microblogs is to include timely, often updated information. Twitter is currently among the most widely used microblogging sites. Since Twitter has such a large and active user base, we decided to focus on tweets for this study. Twitter tweets can be up to 280 characters. It also includes URLs and material like photographs and videos. Users can follow other users' message streams to establish a network. When a Twitter user tweets, it has instantly sent to their followers' feeds.

Due to its informal and immediate character, microblogging services such as Twitter offer a handy means for millions of individuals to connect. Intriguingly, software engineers and end users of software systems are active on Twitter. Their tweets may provide a wealth of information. The tweets might distribute information on, for example, the latest software system features, development methodologies, conferences, defects, issues, solutions, feature requests, and so on. These give a wealth of data instrumental in software development, where several "events" occur regularly. These tweets need to be translated into knowledge to learn from collective intelligence. Given the importance of informal communication in software development projects, it makes sense to investigate how microblogs assist communications during software development activities. (Palakorn Achananuparp, 2011)

More than 30 pieces of information, such as the app version and device type, are appended to bug reports filed through Facebook's mobile apps. Despite the availability of these channels, a growing number of users continue to detail their problems on Twitter and other social media.

One theory puts the answer at a desire to put more of a spotlight on software problems in order to put more pressure on the companies who make the programs. Research shows that new features and flaws may be discovered by scanning Twitter rather than traditional routes like app stores. (Daniel Martens, 2019)

Sentiment Analysis (SA), also known as Opinion Mining (OM), is the computational investigation of people's feelings and sentiments about a topic or object. Individuals, events, or concepts can all be represented by the entity. SA may be broken down into three levels: the document level, the phrase level, and the aspect level. The goal of sentiment analysis performed at the document level is to determine if an opinion document expresses a positive or negative opinion or emotion. The entirety of the document is seen as a single fundamental information unit by it. The goal of using SA at the sentence level is to categorize the sentiment communicated in each sentence. The first thing we need to do is determine whether or not the sentence is objective or subjective. If the sentence is subjective, Sentence-level SA will decide if it reflects favorable or unfavorable opinions and grade it accordingly. The Objective of aspect-level SA is to categorize people's feelings about particular characteristics of things. The first thing that must be done is to recognize the entities and how they manifest. (Walaa Medhat, 2014)

1.1.Context and Motivation

The popularity of mobile devices like smartphones and tablets and the ease with which new apps might be developed have led to an explosion in the app sector during the past five years. This need has sparked competition among app makers that want to reach users wherever they may be. So, developers must have a forum where they can ask for and receive user opinions on what features should be included and how the app may be enhanced. (Maleknaz Nayebi H. C., 2018)

The critical user feedback for analysts and app developers are feature requests and issue reports. As the "voice of the users," this input can help shape future iterations and inform development priorities.

The previous research was conducted by considering the applications featured on the top charts on Google Play (all of the top chart apps across a variety of categories), in addition to random choices of apps that had a range of ratings and numbers of reviews. We chose 70 applications at random to conduct an in-depth content study, during which we compared the content of app store evaluations with the content of tweets connected to app topics. These comparisons highlight how Twitter can supply developers with additional information that cannot be gained through app store ratings. In order to do this, we made use of automatic categorization, and then we moved on to topic modeling. (Maleknaz Nayebi H. C., 2018),

1.2 Problem statement, objectives, and research question

1.2.1 Problem statement:

Tweets may be about everything from pop culture to current events to politics to technology. Users also utilize Twitter to communicate about software programs, as shown through a brief investigation utilizing the search tool provided by Twitter. Because of this, requirements engineers and other stakeholders inside software organizations may find that tweets are a valuable source of information to refer to. In this regard, tweets might be compared to app evaluations written by users and consist of software recommendations, reports on problems, and requests for more functionality. Software firms might better understand their consumers' and users' demands with the assistance of tweets. In addition, they collected data from dispersed and distant consumers, which are often difficult to include. The insights obtained from tweets might then be utilized to make educated judgments within the software evolution processes at cross country level.

1.2.2 Objective:

To perform sentimental analysis on tweets of software support accounts from Twitter and classify their sentiments at the software product level and country level.

1.2.3 Research Goals:

- Explored different Tweepy API endpoints of Twitter to collect the tweets of software support accounts
- Learned to develop a REST client using python to query the Tweepy APIs

- Identified the country-based information of the tweets by extracting geolocation data of each tweet
- Explored the VADER sentimental analysis that is widely used to analyse the text content of social media
- Applied Mann–Whitney U test at country level of the tweet data set

1.3.Overview of the dissertation

We have gathered 15072 tweets from the support accounts of six software programs across three nations over the course of two months for study. This research examines software support accounts' tweets at a national level to ascertain if they are positive, negative, or neutral in tone using VADER sentimental analysis.

2. Literature Review

2.1. Background Information

2.1.1 Twitter

Twitter has quickly become one of the most popular social media platforms. The success of many businesses and public figures depends on their ability to maintain a lively Twitter presence and regularly interact with their followers.

Regarding social capital, Twitter can be thought of as either a pyramid or a triangle. There are major players at the top, like CNN, the New York Times, and TIME, which have millions of viewers who tune in to hear about the latest scandal involving a famous person. These Twitter users with many followers are the cherry on top; they are everyone's first stop for news and updates. That said, it is not a one-way street. For a pyramid to work, its pinnacle must be supported by the levels below it. One's social capital, or "pyramid," is represented by the number of people who follow them on Twitter. (Lyon, 2011)

The information deemed valuable spreads on the microblogging platform Twitter via a series of super hubs, influencers, or alpha users that reach a large audience of attentive and engaged users. Apart from these super hubs where thousands of users retrieve information snippets, pyramid-formed or circular social micro-networks that evolve around each active user also contribute to the spread of information. (Isabel Anger, 2011).

Instead, Twitter has developed into a reservoir of dynamic information streams, including links, brief status updates, and first-hand accounts of breaking news. A handful of these "alpha users" can significantly impact the opinions of the whole community. They have a sizable, engaged following that reads and shares their posts. Therefore, an influencer's material (whether text or links) is disseminated throughout various micro-networks and is seen by many people who may or may not be the influencer's direct followers: the farther the material spreads, the more of an impact the user has.

Keeping our Twitter account updated by posting fresh tweets and retweets, following relevant accounts, and responding fast to direct messages is integral to building a solid online reputation.

Twitter API:

The **Twitter Application Programming Interface** (API) grants programmers access to most of Twitter's features. Tweets, users, and trends are all examples of Twitter entities whose data may be read and written via the API.

Twitter's REST API and streaming API use the HTTP protocol but serve distinct purposes. The REST API enables programmers to submit tweets, see users' followers, search for recent tweets (limited to tweets within the previous week), and send and retrieve messages using standardized methods. 28 Whereas, the streaming API broadcasts information to carefully selected third parties in real-time. Just because someone is technically savvy enough to make a few HTTP calls does not mean they have access to all of the data Twitter stores. (Bucher, 2013)

The Streaming API and the Search API are two of Twitter's available application programming interfaces. Data in real-time may be collected using the Streaming API, while data from the past can be retrieved with the Search API. The Search API is available at three pricing tiers (Standard, Premium, and Enterprise). According to Standard Search API's Documentation, "a selection of recent Tweets written in the previous seven days" is what we may expect to get. Besides Free, not many academics utilize Premium or Enterprise due to their high prices. Real-time Tweet collecting is now possible with the help of a streaming API. There are two alternatives, the Free Streaming API and the Decahose Streaming API, to choose from when the goal is to gather all Tweets, regardless of their subject matter. According to Twitter's instructions, all clients that connect using the free option within the same time window receive the same random sample of all public Tweets. Programmers may utilize the Twitter Free Streaming API to pick tweets containing a given term, hashtag, or geolocation. Decahose Streaming API is a premium service that provides a random 10% sample of Twitter Firehose data in real-time (which contains all Tweets). As an alternative, Twitter offers a premium filtering option (named PowerTrack API) that appears to filter the whole Firehose of Tweets (Alina Campan, 2018).

The Twitter API makes available a wide variety of technical HTTP endpoints, including ones for Tweets, Retweets, Likes, Direct messages, Favorites, Trends, and Media. Every request made to the Twitter API is verified using OAuth, a popular open authorisation standard. We must generate and set up our login credentials before making calls to the Twitter API. Using the Twitter API, programmers may create bots, analytics, and other forms of Twitter automation. Twitter's API also restricts how often We may call certain API functions. If we go over this limit, we will not be able to use the API again for 5–15 minutes. (Garcia, 2022)

Tweepy:

Tweepy simplifies access to the Twitter streaming API by taking care of tedious tasks like authenticating users, establishing connections, establishing and terminating sessions, reading incoming messages, and partially rerouting them. (Roesslein, 2018)

As an open-source Python module, Tweepy provides a simple interface for communicating with the Twitter API. Tweepy's classes and methods stand in for Twitter's models and API endpoints, while the library transparently takes care of implementation specifics like Data encoding and decoding, HTTP requests, Results pagination, OAuth authentication, Rate limits, and Streams (Garcia, 2022)

Tweepy includes different access levels to help scale our platform usage. They are Essential, Elevated, Elevated+, and Academic Research. In this project, we use the essential level access for the Twitter API. Below is the reference of the screenshot for the multiple access levels. (Twitter, 2022)

	Essential	Elevated	Elevated+ (coming soon)	Academic Research
Getting access	Sign up	Apply for additional access within the developer portal	Need more? Sign up for our waitlist	Apply for additional access
Price	Free	Free		Free
Access to Twitter API v2	✓	✓		✓
Access to standard v1.1	✓ (Limited access - only media endpoints)	✓		✓
Access to premium v1.1	✗	✓		✓
Access to enterprise	✗	✓		✓
Project limits	1 Project	1 Project		1 Project
App limits	1 App per Project	3 Apps per Project		1 App per Project
Tweet caps	Retrieve up to 500k Tweets per month	Retrieve up to 2 million Tweets per month		Retrieve up to 10 million Tweets per month
Filtered stream rule limit	5 rules	25 rules		1000 rules
Filtered stream rule	512 characters	512 characters		1024

Figure 1 TwitterAPI account subscription list

The API methods can be grouped into the following categories:

- Methods for user timelines
- Methods for tweets
- Methods for users
- Methods for followers
- Methods for our account
- Methods for likes

- Methods for blocking users
- Methods for searches
- Methods for trends
- Methods for streaming

In the following subsections, we will review different API method groups.

2.1.2 Natural Language Processing(NLP):

Natural language processing (NLP) emerged in the 1950s as a field where AI and linguistics converged. Text information retrieval (IR) uses highly scalable statistics-based approaches to index and search enormous quantities of text quickly, and NLP was initially seen as separate from IR. (Prakash M Nadkarni, 2011)

In order to "achieve human-like language processing," as indicated above, NLP is being developed. There is much thought that went into picking the word "processing," therefore the more generic "understanding" would be a poor substitute. For a while, NLP was formerly known as Natural Language Understanding (NLU) in the early days of artificial intelligence. It is generally accepted that while real NLU is the ultimate aim of NLP, this goal has not yet been achieved. What a complete NLU System could do First, Take an existing text and rephrase it. Second, Have it translated into another language. Third, discuss the material presented in the book. Fourth, deduce what the text means. (Liddy, 2001)

Natural language processing (NLP) integrates statistical, machine learning, and deep learning models with computational linguistics (the rule-based modeling of human language). When combined, these tools provide computers the ability to 'understand' human language, in the form of text or audio data, in its entirety, including the speaker's or writer's purpose and mood. Computing applications powered by natural language processing (NLP) can now translate across languages, respond to voice instructions, and quickly summarize vast amounts of text, often in real-time. We may have already engaged with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other modern conveniences. However, natural language processing (NLP) is now playing an increasingly important role in corporate solutions that aim to improve the efficiency of businesses, boost employee productivity, and simplify crucial business procedures. (IBM Cloud Education, 2020)

Due to the inherent ambiguities of human language, it is exceedingly challenging to develop software that can correctly ascertain the intended meaning of text or speech data. Programmers must teach natural language-driven applications to recognize and understand accurately from the outset the idiosyncrasies of human languages, such as homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, and variations in sentence structure.

Several natural language processing tasks deconstruct human text and speech input in ways that are understandable to a machine. The following are examples of such responsibilities:

Any program that responds to voice commands or questions will need to be able to recognize human speech. In *speech recognition*, also known as a speech-to-text conversion, accuracy in turning audio into text is a top priority. A significant obstacle to perfect speech recognition is the way people speak: rapidly, with slurred words, varying emphasis, and intonation, in a variety of accents, and frequently with incorrect grammar.

Part of speech tagging Tagging a word or passage with its grammatical function based on how it is typically used is an example of part of speech tagging, also known as grammatical tagging. I can create a paper airplane, but what make of automobile do we drive?" "make" is a verb when it describes an action, and it is a noun when it describes a type of vehicle.

Word sense disambiguation is using semantic analysis to select the most appropriate meaning of a word that can have more than one. Word sense disambiguation can be used to tell the difference between the two uses of the verb "make," such as "make the grade" (achieve) and "make a bet" (put a wager) (place).

The *Named entity recognition* technique recognizes vital concepts embedded in the text. A person named "Fred" or the state of "Kentucky" are both things that NEM can recognize.

Co-reference Resolution, we must determine whether or not two terms refer to the same thing. Most often, this means figuring out who or what a pronoun refers to (e.g., "she" = "Mary"), but it can also mean spotting a metaphor or idiom (e.g., "bear" isn't an animal but a large hairy person).

Sentiment analysis aims to identify and understand the underlying attitudes, emotions, sarcasm, bewilderment, or mistrust in written content.

Sometimes viewed as the polar opposite of voice recognition and speech-to-text systems, *natural language generation* entails translating formal data into understandable, conversational language.

2.1.3 Sentimental Analysis:

Opinion, judgment, and emotion may be extracted from text using sentiment analysis. Analyzing a piece of written or spoken text's positive, negative, or neutral tone is called sentiment analysis. As a result, it provides a helpful indicator of the customer's overall satisfaction. Our free-form language would be converted into categorized information (such as "positive," "negative," or "neutral"), aggregated with the opinions of many others, and summarized to provide a business with a high-level picture of the public's reaction to the brand or product in question. To better understand the social sentiment of a brand, product, or service, businesses might employ sentiment analysis, contextual mining of text that finds and extracts subjective information in the source material. (Qualtrics Blog, 2022)

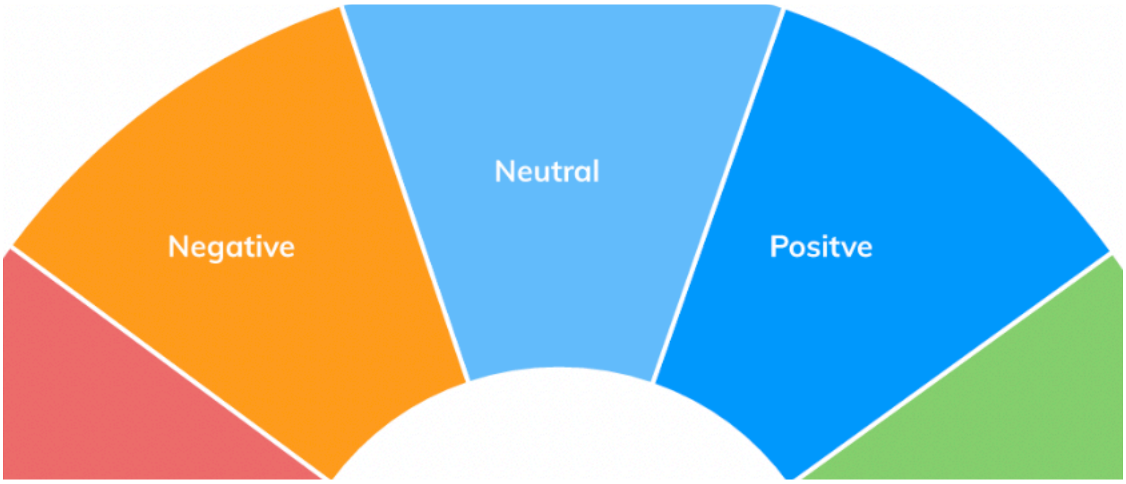

Figure 1 Category information of sentiment

Sentiment analysis is a subset of text mining (sometimes called text analysis) that aims to glean meaning from unstructured material. Meaning is gleaned from a wide variety of texts using this method, including online polls, reviews, public social media posts, and online articles. Afterward, the text's emotional tone is used to provide a score. A score of -1 would indicate pessimism, whereas a score of +1 would indicate optimism. NLP (natural language processing) is utilized for this purpose (NLP). Below is the example image for plotting the analyzed text's positive, negative, and neutral sentiments. (Qualtrics Blog, 2022)

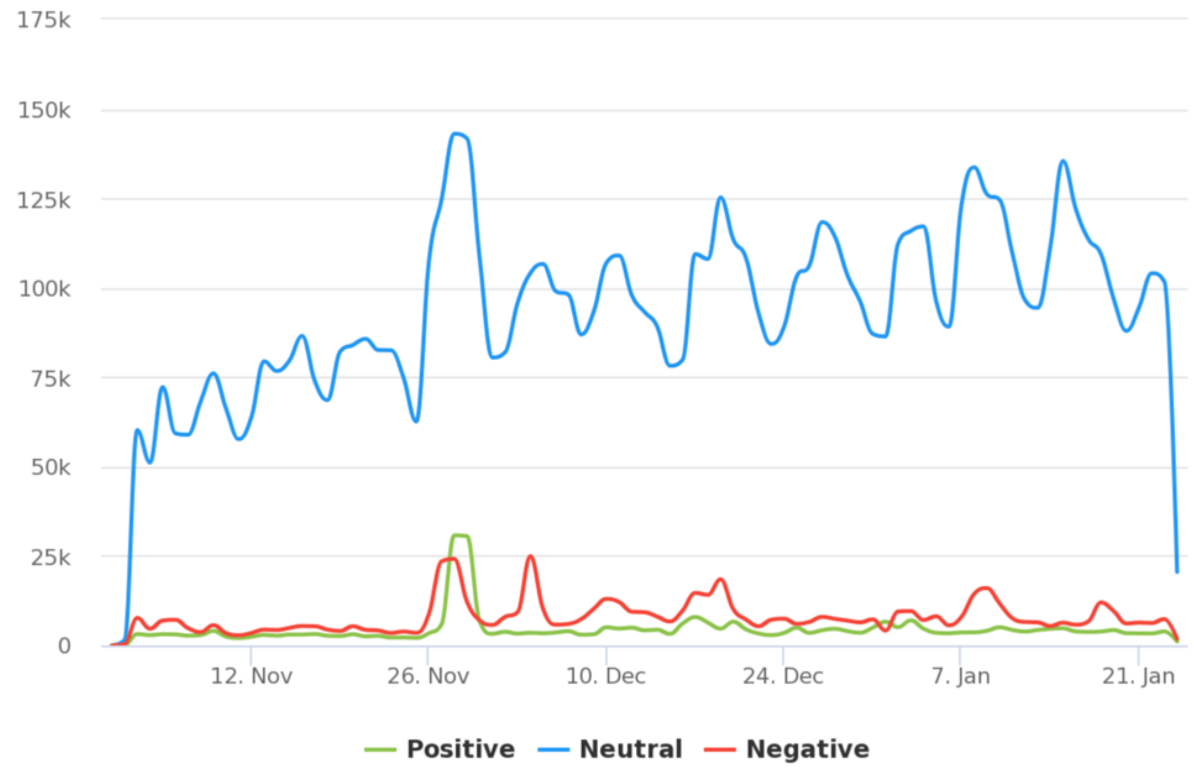

Figure 2 Graphical Representation of sentiments

When we need to draw broad conclusions from a vast amount of textual data, sentiment analysis can help.

Say we are in the marketing for a large movie company and just released a teaser that generated many tweets. We can read some or many comments, but we will not know how many people liked or hated it unless we read every single one and record whether it was positive, negative, or neutral. That would be prohibitively expensive, time-consuming, and error-prone. The individual or people reading the comments might also be biased. Their views or perceptions may color how they interpret the data, and their judgment may change based on mood, energy levels, and other customary human variations. With sentiment analysis tools, we can get a quick, consistent verdict. Sentiment scores help companies understand how their brand affects consumers. Happiness, sadness, anger, or neutrality are examples. The firm must next decide how to implement this emotion. Let sentiment guide how customers experience our brand. Sentiment research reveals how customers see our brand. Customer feedback through social media, the website, service agents, or other sources includes essential business information, but it is not enough to know what customers are talking about. Knowing how people feel will reveal their experiences. Sentiment analysis can help with this. Sentiment analysis can tell us if public perception about our business has changed. Peaks or dips in sentiment ratings help us enhance products, educate salespeople or customer service agents, or build new marketing initiatives. (Qualtrics Blog, 2022)

Today, there is an enormous increase in the number of 'sentiments' that can be found on social media platforms like Twitter and Facebook, as well as on message boards, blogs, and user forums. These little pieces of text are a gold mine for businesses and people who want to monitor their reputation and gain quick feedback on their products and actions. These firms now have the capacity to monitor the various social media sites in real-time and respond appropriately, thanks to the application of sentiment analysis. The immediate benefactors of sentiment analysis technologies include marketing managers, public relations agencies, campaign managers, politicians, equities investors, and internet shoppers. (Feldman, 2013)

Internet, news stories, social media, and digital communications create and disseminate massive volumes of textual material every instant. Sentiment research may help firms track how their brands and goods are regarded over time. (Qualtrics Blog, 2022)

There are many names for sentiment analysis. Analysis of subjectivity, opinion mining, and the extraction of appraisals are common names for this process, which also has some ties to affective computing (computer recognition and expression of emotion). A sentiment that appears in the text comes in two flavors: explicit, where the subjective sentence directly expresses an opinion ("It's a beautiful day"), and implicit, where the text implies an opinion ("The earphone broke in two days") (Mejova, 2009)

(Qualtrics Blog, 2022)Depending on the application and desired output, a variety of strategies and algorithms can be employed to accomplish the task at hand. Examples of simple forms of sentiment analysis are:

*Detecting sentiment***:** This requires separating subjective information (such as "I love this!") from neutral facts (such as "the restaurant is located downtown").(Qualtrics Blog, 2022)

*Categorising sentiment***:** This involves identifying if the tone is positive, negative, or neutral. The tool-provided weighting of these gradations—from highly positive to merely neutral to merely negative—is possible. (Qualtrics Blog, 2022)

*Clause-level analysis***:** Sometimes, literature offers divided or ambiguous sentiments, for example, "staff were really friendly but we waited too long to be served." Assessing feedback at the clause level demonstrates when both excellent and poor opinions are stated in one location and might be valuable in case the positives and negatives inside a text cancel each other out and yield a false "neutral" result. (Qualtrics Blog, 2022)

How consumers "feel" about our business might be a valuable metric for gauging client satisfaction in today's age of infinite online comments. Consumers are looking for businesses they can connect with and trust online and offline. If a customer is emotionally invested in our product, they are more inclined to express those feelings in the text (through surveys, reviews, social media, and more). That being said, the inverse is also true. 71% of Twitter users will utilize the service to air their grievances about a company. Both positive and negative customer feedback should be recorded and analyzed to enhance future service. We can benefit from sentiment analysis. Let us start with analyzing more immediate interactions with brands. (Qualtrics Blog, 2022)

Social media mining is a relatively unobtrusive technique for gathering textual information. Monitoring features included in social media management tools make this possible. Social media monitoring systems, in essence, crawl public platforms like Twitter and Facebook for mentions of brands, then assign sentiment ratings to such comments. It has benefits, too, because people will likely share their honest opinions on social media. The character count restriction of social media posts is a drawback of text analysis. For example, Twitter has a tweet content limitation of about 280 characters. Brands may be losing out on valuable consumer feedback because of the sheer volume of information available on social media platforms that has yet to be analyzed. (Qualtrics Blog, 2022)

2.1.4 VADER sentimental Analysis

VADER stands for Valence Aware Dictionary and sEntiment Reasoner. VADER is a lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in social media. VADER is a basic rule-based model for broad sentiment analysis. It is compared against eleven state-of-the-art benchmarks, including LIWC, ANEW, the General Inquirer, SentiWordNet, and machine learning approaches using Naive Bayes, Maximum Entropy, and Support Vector Machine (SVM) algorithms. This technique is widely used for sentiment in microblog-like contexts. It uses qualitative and quantitative techniques to develop and test a sentiment lexicon explicitly tailored to the language of microblogs. Next, we integrate these linguistic characteristics with five generalizable criteria individuals utilize to convey or emphasize sentiment intensity. These heuristics contribute to an increase in the accuracy of the sentiment analysis engine in a variety of different domain contexts (C.J. Hutto, 2014).

The VADER lexicon has a remarkable amount of success in social media. The VADER model is a less resource-intensive approach to opinion mining that specifies a mathematical model by applying a set of rules instead of directly coding it. When compared to Machine Learning models, VADER requires a significantly less quantity of training data, which results in a reduction in the number of resources it needs to run. The efficient method VADER uses allows us to decode and measure the emotions in the text, audio, and video streams in real time. VADER's speed versus performance tradeoff is hardly noticeable with VADER (C.J. Hutto, 2014).

When we take a closer look at the F1 scores, which measure the accuracy of classification, we find that VADER (0.96) performs much better than individual human rates (0.84) in terms of adequately classifying the sentiment of tweets into positive, neutral, or negative categories.

The explanation is that VADER is sensitive to the polarity (whether the feeling is positive or negative) and the intensity (how positive or negative the feeling is). VADER considers this and incorporates it into its system by assigning a Valence Score to the word. (Calderon, 2017)

Valence Score and its calculation:

A valence score is a score that is awarded to the word in question based not on pure logic but observation and experiences rather than on logic alone. Take the adjectives "awful," "hopeless," and "miserable," for example. Any self-aware human being would immediately recognize that the tone conveyed by these remarks is unfavorable. On the other hand, adjectives such as "marvelous," "worthy," and "adequate" are indicative of a positive attitude. According to the research study conducted on VADER, the Valence score is calculated using a scale ranging from -4 to +4, with -4 being the most "Negative" emotion and +4 representing the most "Positive" sentiment. It is possible to deduce from common sense that the point of 0 in the middle indicates a 'neutral' sentiment; this is how it is described as well (Swarnkar, 2020).

(Swarnkar, 2020)VADER uses a lexicon that maps words in addition to the various other lexical elements common to expressing sentiment in microblogs. These characteristics are as follows:

- A complete index of emoticons written in the Western script (for example - :D and :P) (Swarnkar, 2020)
- Acronyms that are connected to sentiment (for example - LOL and ROFL) (Swarnkar, 2020)
- A form of vernacular speech that conveys an emotional meaning (for example - Nah and meh) (Swarnkar, 2020)

(Swarnkar, 2020)It is time-consuming and potentially error-prone to compile a comprehensive lexicon of feelings and emotions manually. Consequently, it should be no surprise that a significant number of Natural Language Processing (NLP) researchers place a heavy emphasis on existing dictionaries as crucial resources.

(Swarnkar, 2020)In order to take into account the influence of each sub-text on the seeming gravity of the mood conveyed by the sentence-level text, VADER employs a set of criteria. Heuristics is the name given to these guidelines. There are a total of five of them.

(Swarnkar, 2020)The use of *punctuation*, precisely the exclamation mark (!), raises the level of intensity without altering the meaning of the sentence's direction semantically. As an illustration, "The weather is hot!!!" is more emphatic than the expression "The weather is hot."

(Swarnkar, 2020)The use of *capitalization*, and more especially utilizing ALL CAPS to emphasize a sentiment-relevant word in the presence of other words that are not capitalized, raises the magnitude of the intensity of the feeling without changing the semantic orientation. For instance, saying "The weather is HOT." rather than "The weather is hot." gives a stronger sense of urgency.

(Swarnkar, 2020)The influence of *degree modifiers*, also known as intensifiers, booster words, or degree adverbs, on the intensity of a feeling can take the form of either an increase or a decrease in intensity. For instance, the phrase "The weather is extremely hot" conveys a higher level of intensity than the phrase "The weather is hot," yet the phrase "The weather is mildly hot" conveys a lower level of intensity.

(Swarnkar, 2020)*Polarity shift due to Conjunctions*, The contrasting conjunction "but" denotes a switch in the polarity of the expressed feeling, with the sentiment of the text that comes after the conjunction being the more prominent one. For instance, the statement "The weather is hot, yet it is manageable." conveys a range of feelings, with the second half of the sentence being the primary factor determining the total rating.

(Swarnkar, 2020)*Catching Polarity Negation*, Examining the consecutive sequence of three items that comes before a lexical characteristic loaded with emotion allows us to catch over 90 percent of the instances in which negation changes the polarity of the text. A sentence like "The weather is not actually that hot." is an example of a sentence that uses negation.

SentimentIntensityAnalyzer() is an object, and *polarity_scores* is a method that will give us scores in the following categories: Positive, Negative, Neutral, and Compound. (Kung, 2021)

The compound score is the sum of positive, negative & neutral scores, normalized between -1(most extreme negative) and +1 (most extreme positive). The closer the Compound score to +1, the higher the positivity of the text. VADER has been ported to Java, JavaScript, PHP, Scala, C#, Rust, and Go. (Kung, 2021)

Using the compound scores supplied by the algorithm, we have categorized the gathered tweets from the Tweepy APIs as either positive, negative, or neutral based on the VADER sentimental analysis approach.

Both TextBlob and VADER use lexicon-based methods. Sentiment analysis using a lexicon relies on a mapping between words and emotions, with a sentence's overall sentiment being the sum of its terms. VADER is social media-oriented, which is where it differs most from TextBlob. As a result, VADER invests significant time and energy in recognizing the emotions conveyed by common social media content elements like emojis, repeated sentences, and punctuation (exclamation marks, for example) (White, 2020).

2.1.5 Mann-Whitney U Test:

(Marsja, 2020)When the dependent variable is ordinal or continuous but not normally distributed, the Mann-Whitney U test assesses the differences between the two groups. For instance, the Mann-Whitney U test can help us figure out if there are significant gender differences in people's opinions on wage discrimination if such opinions are assessed on an ordinal scale (i.e., our dependent variable would be "attitudes towards pay discrimination" and our independent variable would be "gender," which has two groups: "male" and "female"). The Mann-Whitney U test can be used instead to see if there is a significant difference in earnings after controlling for factors such as degree of education (i.e., our dependent variable would be "salary" and our independent variable would be "educational level," which has two groups: "high school" and "university"). The Mann-Whitney U test is frequently (but not always) used as a nonparametric alternative to the independent t-test. Different results can be drawn from the Mann-Whitney U test than the independent-samples t-test, depending on the assumptions about the data's distribution. Conclusions might range from merely asserting that the two populations are distinct to identifying disparities in medians across groups. We will elaborate on why the form of our data distributions matters for drawing these various inferences.

(Marsja, 2020) The Mann-Whitney U test is a nonparametric test of the null hypothesis that the distribution underlying sample x is the same as the distribution underlying sample y. It is often used to test the difference in location between distributions.

(Marsja, 2020) Requirements for applying Mann-Whitney U test

- (Marsja, 2020) Two random, independent samples

- (Marsja, 2020) The data is continuous

- (Marsja, 2020) The scale of measurement should be ordinal, interval, or ratio

- (Marsja, 2020) For maximum accuracy, there should be no ties, though this test - like others - has a way of handling ties

(Karadimitriou, 2018)The Mann-Whitney U test determines if two independent groups vary with respect to a given dependent variable. The test examines if the two groups come from the same population by comparing whether or not they have the same distribution of the dependent variable. For each group, the test first ranks all the dependent variables (the lowest value is assigned a rank of 1) and then utilizes the total of these rankings to determine the test statistic.

It is possible to conceive of a scenario in which a researcher has two groups of subjects but relatively few people in each group (less than eight participants). Therefore, due to a lack of data, this researcher cannot conclude that his two groups are normally distributed. Data collected by this experimenter are of continuous or ordinal type, and there is a statistical "constraint" limiting the data's variability. It suggests that there may be some margin for error in his measurements. Therefore, the researcher cannot apply the parametric test of the mean using the Student's t-distribution, as it is not feasible to determine whether or not the two samples are normally distributed. What is the proper response in this kind of predicament? At first, the researcher is forced to use a nonparametric statistical test (nonparametric tests are required when the distribution is asymmetrical). Compared to parametric tests, nonparametric tests allow the model structure to be decided by the data rather than described in advance. The word "nonparametric" does not indicate that these models have no parameters; instead, it refers to the fact that both the quantity and type of parameters can vary. Because of this, nonparametric tests are sometimes referred to as distribution-free tests. It is possible to address the researcher's inquiries about the differences between his groups by using the Mann-Whitney U test. This study's strongest suit is its potential applicability to samples as small as five to twenty people. This method was applied when ordinal data were recorded using a subjective and not exceptionally exact scale. (Nachar, 2008)

3. Related Work:

Daily, millions of tweets cover a wide range of subjects, all posted by users of the Twitter microblogging network. Some of these tweets' content is useful for software industry businesses and might be used by, for instance, requirements engineers to better understand what end users want. Twitter is used to communicate about software applications and to understand the prospect of using tweets to inform requirements engineering and software evolution tasks. They looked into how software applications are discussed on Twitter, the substance of those tweets, and the possibility of automating the process of filtering out information relevant to those discussions. The term "usage" refers to how Twitter users discuss different kinds of software. (Emitza Guzman R. A., 2016)

A Mobile app is a piece of software designed to run on a mobile device. Once an app has been created, it may be distributed and sold through an app store. The production of apps is motivated mainly by consumer demand. The needs for an app are typically drawn from strategic corporate objectives or market possibilities, much like with more traditional market-driven software. App creators seldom interact with consumers during the early stages of development. The number of app downloads and money made from them is the two primary indicators of success. Because of the widespread availability of app stores, formerly exclusive software was no longer required to reach a global audience. Although there are many positives to app shops, there are also many difficult ones. Apps might fail (get few or no downloads) in today's crowded and competitive app market for reasons that have nothing to do with their usefulness or ease of use, such as the app's name, icon, or degree of publicity. With the low-profit margins associated with app sales, a product must find widespread adoption throughout the globe. App downloads can be affected by user behavior and country factors, yet many developers are unaware of these differences. The significance of elements like the app's description, images, price, and user feedback is poorly understood. Many apps have failed due to these problems. (Soo Ling Lim, 2015)

With the proliferation of smartphones and tablets has come an explosion in the size of the app store industry, which has piqued the interest of several academic and business platforms interested in app store mining. Reviews from the app stores are used to examine several facets of app growth and development. However, comments from app users may be found elsewhere outside the app store. In reality, the significance and value that social media postings might give in the context of mobile app development go largely untapped despite the sheer volume of posts that are created every day. Twitter's supplementary data can help with mobile app enhancements. They also discovered that, for the most part, the number of reviews and tweets about an app are highly correlated. (Maleknaz Nayebi H. C., 2018)

Research into Sentiment Analysis (SA) is a dynamic subfield of text mining. SA refers to using computers to process the subjective nature of the text, including its views and feelings. It gives us a synopsis of the most recent developments in the subject. Several newly proposed algorithm improvements and a wide range of SA applications are explored and briefly described in this overview. Each item is arranged into a category based on its significance to a particular type of SA method. (Walaa Medhat, 2014)

(Emitza Guzman M. I., 2017)This research shows that tweets may be a valuable source of information for developing software and its requirements, including details about user needs and complaints about bugs and other issues. Because of this, Twitter has become a vital tool for requirements engineering and software evolution that relies on input from a large group of users. Due to the sheer volume, lack of organization, and inconsistent quality of tweets, human analysis is just not viable. In order to do things like summarizing, categorizing, and ranking tweets, automated analysis techniques are required. Twitter is one of the most popular social media platforms since it has an average daily volume of more than 500 million micro-messages, generally known as tweets. A previous study indicated that users of Twitter compose tweets regarding software applications. These tweets contain helpful information that may be utilized to drive software evolution and elicit new needs, such as feature requests, problem reports, and feature deficiency descriptions.

Through micro-blogs, Twitter allows huge populations of software end-users to discuss their experiences and complaints regarding various software systems openly. These types of data may be gathered and categorized to assist software developers in inferring the demands of users, discovering flaws in their code, and planning for future versions of their systems. Nevertheless, the process of automatically acquiring, categorizing, and displaying meaningful tweets is not a simple one. Problems arise due to the vast quantity of available data, its singular structure, varied composition, and a high proportion of useless information and spam. Text classification techniques such as Support Vector Machines and Naive Bayes can be instrumental in certain situations. Also, it is efficient in identifying and classifying information that is technically relevant tweets taken from the total number of gathered tweets. (Grant Williams, 2017)

4. Methodology

4.1. Overall Methodology:

This project aims to analyze sentiment on the tweets of software product support accounts such as Facebook, Instagram, Snapchat, WhatsApp, Telegram, and Messenger using VADER (Valence Aware Dictionary and sEntiment Reasoner). Tweepy APIs are used to gather the tweets, and a generic python script designed for this purpose is used for any software account. Any software product's support Twitter account can use the coded logic to retrieve tweet data. Understanding the tone of the phrases used in tweets is aided by the presentation of data using various python libraries. Also, the Man Whitney U test was used to assess the dissimilarities between software product and location pair data.

4.2. Data collection and analysis:

We acquired the tweet data of software product support accounts using Tweepy APIs. There are several variations of the Tweepy API, and the Python version is the one that is utilized here. Because the Twitter API is guarded by open authorization protocol, we need to have particular API keys to access their APIs.

Steps to collect the tweets from the Tweepy APIs:

To begin collecting tweets, we must, of course, first create a Twitter account. Here is a snapshot of my project-related Twitter feed. My username is "@udhay ss," which we can see above. As Twitter forbids using the same username for several accounts, each user's will be unique.

Figure 3 Twitter account

After the Twitter account has been successfully created, we will need to register with the Twitter developer portal as soon as possible. Many self-service tools available through the Twitter developer portal allow programmers to control their usage of the Twitter API and the Twitter Ads API. We may do the following in the portal:

- Develop and handle our Twitter-related endeavors (and the authentication keys and tokens they provide).
- The Twitter API premium endpoints v1.1 and v2 allow us to control access and manage integrations.
- Discover the many options for terminals and features.
- With Elevated permissions and an organization account, we may visit team pages where we can add and manage the many handles that have access to our team's account.

Before registering for the developer portal, the required registration form needs to be completed on its whole. Before registering for the Twitter developer portal, we must first log in to our Twitter accounts. It allows the Twitter developer portal to automatically read our email addresses, account names, countries, and use cases, as seen in the image below.

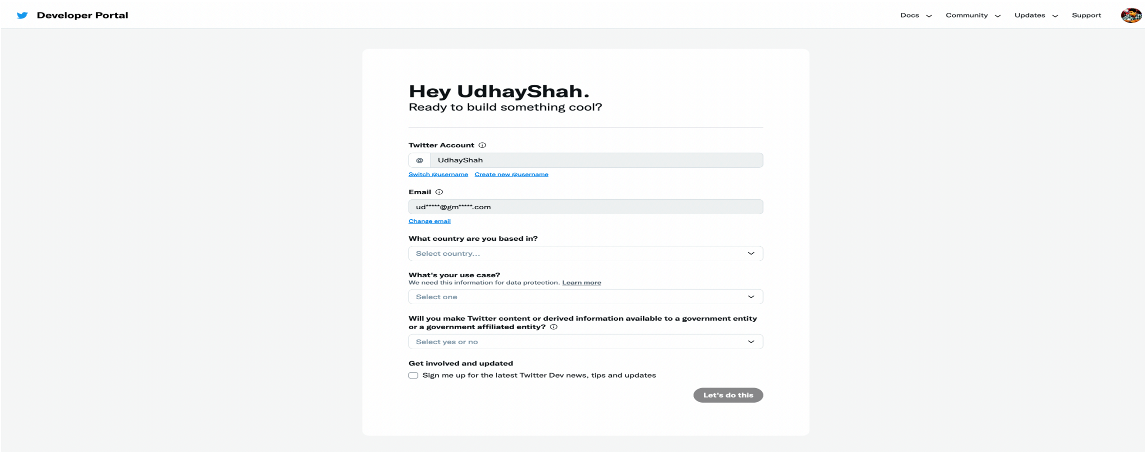

Figure 4 Sign up for a developer portal

In order to establish an account, we will also need to approve the developer agreement and the policy.

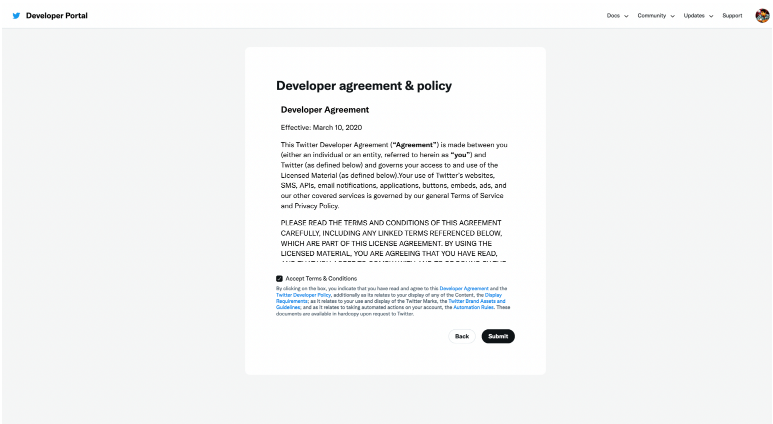

Figure 5 Developer portal Agreement

In order to authenticate ourselves, a verification email will be sent to the email address associated with our Twitter account.

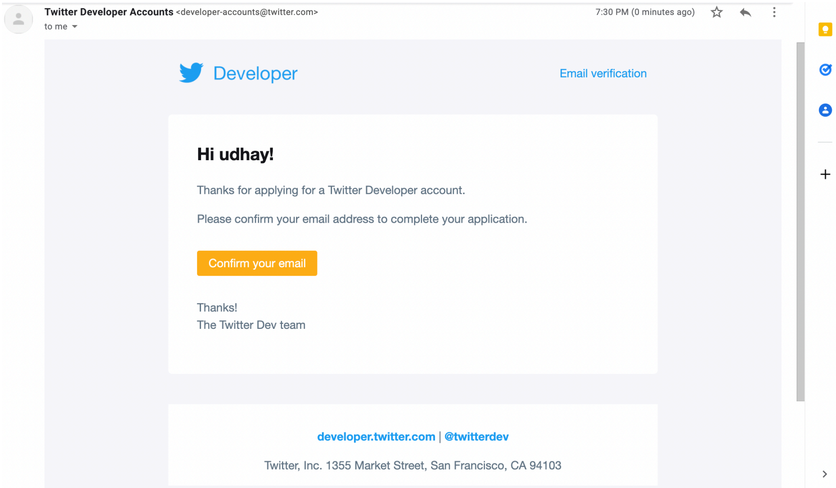

Figure 6 Verification email for the developer portal

After these procedures, the developer account will be established. It asks for a unique app name.

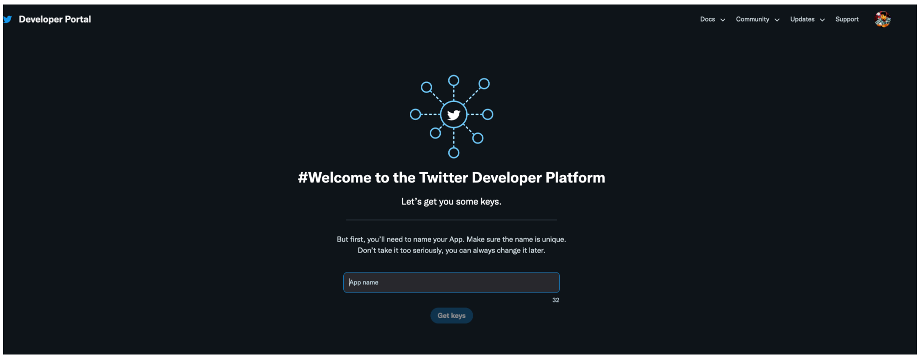

Figure 7 Application name creation

The API key, Secret, and Bearer token for an app are automatically produced once the name is created.

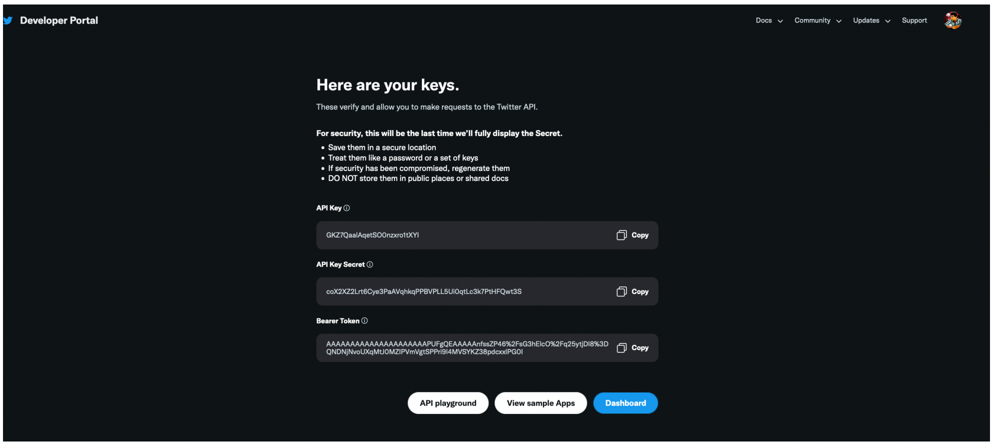

Figure 8 API key generation

We have opted for "Essential" access to the developer portal in this project. The accompanying illustration demonstrates that, with the Essential Account Access plan, we may retrieve up to half a million tweets per project and month at no extra cost.

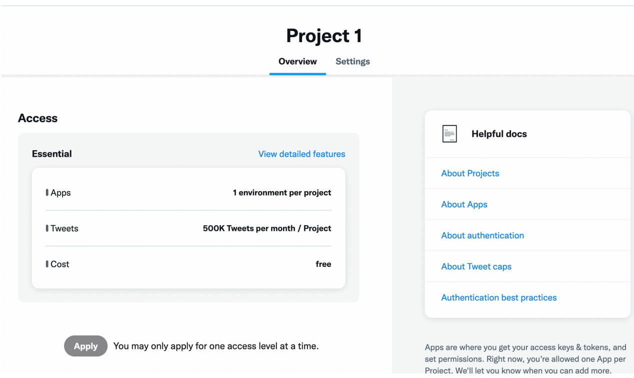

Figure 9 Application name in our portal

First, we need to install Tweepy API on our machine by using the command.

Pip install tweepy

In essence, we have created two different scripts. First, to retrieve the user id for the specified account name of the software product support and a second script to retrieve the tweets of the referenced software product support account.

The scripts used below are written and uploaded to the figshare

https://figshare.com/articles/thesis/Python_script_for_extracting_tweets/20595018

Script 1:

It is hitting the below endpoint of the Tweepy API to fetch the user if for the given username.

https://api.twitter.com/2/users/by/username/:username

For authentication and authorization of the endpoints, we need to attach the generated API token as a header before invoking the method **client.get_user()**.

Users > udhaysubramanian > Documents > myproj > collectdataset.py > ...

```python

import tweepy
import os
from datetime import datetime, timedelta


API_KEY = 'xbSV9lslklEJL0WgKCLry9Moe'
API_KEY_SECRET = 'EKtJxEMeayuAPwGUdJvgslS4ChkrtHqvAsGpGbfmRyzj9ARTK1'
BEARER_TOKEN = 'AAAAAAAAAAAAAAAAAAAAAAPrcwEAAAAATFz6hzNYHoLC8eZwXTfpfg6b91g%3DWFo7w5XAhKraUHMjVcpZFn5myhecOaUVdk2004ZsfKKmAFBRiY'
ACCESS_TOKEN = '1529447481024688129-yfjTQQLTchLruZ3gwL378LUq30aPW7'
ACCESS_TOKEN_SECRET = 'qtLThwT81Yacbi6LMXr0FjLRP9XfS1SWkY0LTwDGeEBR8'

screen_name = 'facebook'

client = tweepy.Client(bearer_token=BEARER_TOKEN, consumer_key=API_KEY, consumer_secret=API_KEY_SECRET, access_token=ACCESS_TOKEN, access_token_secret=ACCESS_TOKEN_S
#print(client)
user = client.get_user(username=screen_name)
#Retrieved the user id using the username(twitteraccountname)
print(user.data.id)
user_id = user.data.id
```

Figure 10 Script 1 logic

From the below image, it is clear that we will be getting the corresponding user id for the given account name of the software product support account.

```
● (base) udhaysubramanian@Udhayas-MacBook-Air myproj % python collectdataset.py
  username: facebook
  1174426031010152448
○ (base) udhaysubramanian@Udhayas-MacBook-Air myproj %
```

Figure 11 Script 1 Sample output

Script 2:

The user id associated with the software product's support account will be sent to the URL below using this script. Each app user is limited to making 450 requests to the below endpoint within 15 minutes.

https://api.twitter.com/2/users/ 1174426031010152448/mentions

The tokens for the API have been appended to the headers, as shown in the figure below. Also, "query params" is a variable that will be provided in GET queries made by REST clients. request().

An additional python list variable, "field names," contains the dataset.csv file's header values.

The loop included inside this script has been iterated until the response is either None or is not equal to the next token. We issued a request to the user mentions endpoint to receive the answer based on the query parameters, which contained the user id and username of the person who tweeted to the software product support account and the person's location, tweet text, and firm name.

The response is also comprised of JSON key-value pairs, to which we have added some criteria to filter out the particular data required for the dataset. In addition, we have included a condition to determine whether or not the response is erroneously terminated when the maximum number of tweets that may be fetched is reached. These retrieved values are compiled into dictionaries of lists, and a CSV file is created with the results.

```
Users > udhaysubramanian > Documents > myproj > test.py > ...
from pip._vendor import requests
from types import SimpleNamespace
import time
import csv

url = 'https://api.twitter.com/2/users/1174426031010152448/mentions'


queryparams={
    "expansions":"author_id,entities.mentions.username,geo.place_id,in_reply_to_user_id,referenced_tweets.id,referenced_tweets.id.author_id",
    "user.fields":"created_at,description,entities,id,location,name,profile_image_url,url,username",
}

headers = {
  "accept": "application/json",
  "authorization": "Bearer AAAAAAAAAAAAAAAAAAAAAPrcwEAAAAATFz6hzNYHoLC8eZwXTfpfg6b91g%3DWFo7w5XAhKraUHMjVcpZFn5myhec0aUVdk2004ZsfKKmAFBRiY"
}

fieldnames = ['text', 'author_id', 'id', 'username','location']

# csv data
filename = "/Users/udhaysubramanian/Documents/dataset.csv"

response = None
next_token = None
while response is None or next_token:
  response = requests.request("GET", url, headers=headers, params=queryparams).json()
  list1=[]
  list2=[]
#print(response)


  for key, value in response.items():
    print("kt",key)
   # print(type(value))
   #print(value)
    seconddict = {}
    if value == "429" or value == "Too Many Requests":
      print("time is currently at an interval of 15!")
      print(next_token)
      print(value)
      time.sleep(15 * 60)
    if key == "errors":
      print("eror")
```

Figure 12

Users > udhaysubramanian > Documents > myproj > test.py > ...

```
        if key == "errors":
            print("eror")
            print(value)
            response = requests.request("GET", url, headers=headers, params=queryparams).json()
        if key == "data":
            print(type(value))
            for index in range(len(value)):
                c=0
                firstdict = {}
                for key1 in value[index]:
                    #print(key1)
                    if key1 == "id" or key1 == "text" or key1 == "author_id":
                        #print(value[index][key1])
                        c+=1
                        firstdict[key1] = value[index][key1]
                        if c == 3:
                            list1.append(firstdict)
                            firstdict = {}
                            c=0
        if key == "includes":
            for k1,v1 in value.items():
                for index in range(len(v1)):
                    c1=0
                    seconddict = {}
                # print(v1[index])
                    for key2 in v1[index]:
                        if key2 == "id" or key2 == "username" or key2 == "location":
                            c1+=1
                            seconddict[key2] = v1[index][key2]
                            if(c1 == 3):
                              #print(seconddict)
                                list2.append(seconddict)
                                seconddict={}
                                c1=0
        if key == "meta":
            #print("meta")
            for k3,v3 in value.items():
                if k3 == "next_token":
                    next_token = v3
                    #print("v3",v3)
        for val1 in list1:
            #print(type(val1["author_id"]))
            for val2 in list2:
                #print(type(val2["id"]))
```

Figure 13

```
Users > udhaysubramanian > Documents > myproj > test.py > ...
        for val1 in list1:
            #print(type(val1["author_id"]))
            for val2 in list2:
                #print(type(val2["id"]))
                if(val1["author_id"] == val2["id"]):
                    #print("kenannn")
                    val1["username"] = val2["username"]
                    val1["location"] = val2["location"]
    if next_token:
        #print("nexttoken")
        queryparams={
       # "pagination":next_token,
        "expansions":"author_id,entities.mentions.username,geo.place_id,in_reply_to_user_id,referenced_tweets.id,referenced_tweets.id.author_id",
        "user.fields":"created_at,description,entities,id,location,name,profile_image_url,url,username",
        #"max_results":"100"
        }
    else:
        queryparams={
        "expansions":"author_id,entities.mentions.username,geo.place_id,in_reply_to_user_id,referenced_tweets.id,referenced_tweets.id.author_id",
        "user.fields":"created_at,description,entities,id,location,name,profile_image_url,url,username",
        #"max_results":"100"
        }


    #print(list1)
    with open(filename, 'a', encoding='UTF8', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(list1)
    # temp = temp + 1
    # print("temp",temp)
    #if temp >= 50000:

```

Figure 14

Fig 14,15,16 logic for the script 2

The completed data set will resemble the picture below. *Fig 17 output of script 2*

	A	B	C	D	E	F
1	Text	Author_id	Id	Username	Location	Account
2	@boredowntw @snapchatsupport yea i just contacted customer support on there website. idk if its a malfu	1.34E+18	1.55E+18	PENIELMARAJ	□üá¨□üá≠ || □ùêõ□ùê•□ùê§ || □ùê°	snapchat
3	@alexolix05 @lostsoulnw @snapchatsupport T,Äôas de la chance, moi impossible de ne serait-ce me conn	1.54E+18	1.55E+18			snapchat
4	@snapchatsupport I stil can,Äôt log in to my account it says bc of to many wrong attempts my account had	1.40E+18	1.55E+18	gamermika19	Roermond, Nederland	snapchat
5	@snapchatsupport @snapchatsupport @Snapchat I have not been received an email or anything to help m	1.55E+18	1.55E+18			snapchat
6	@snapchatsupport Help i have been having this issue since yesterday i can,Äôt even text family or my kids ,	1.47E+18	1.55E+18			snapchat
7	Hello??!!! @snapchatsupport @Snapchat @SnapforDevs I,Äôve been tweeting you and no response for like	1.43E+18	1.55E+18			snapchat
8	@snapchatsupport @snapchatsupport how long is temporary	1.48E+18	1.55E+18			snapchat
9	@sean_deeks @ceriannethomas @Snapchat @snapchatsupport Exact problem I'm having now and yes ear	344956441	1.55E+18			snapchat
10	@oGHisses @snapchatsupport Mine won't work either and I ahe just S22	1.45E+18	1.55E+18			snapchat
11	@snapchatsupport when switching between cameras on my s22, i cant switch to frontfacing camera, but it	1.09E+18	1.55E+18	DrMagnusW1	Steinkjer, Norge	snapchat
12	@snapchatsupport Mine still ain,Äôt working and y,Äôall couldn,Äôt help me via DM nor email,Ä¶ □ü§¶□üè	1.12E+18	1.55E+18	lexiLeighaHart	Florida, USA	snapchat
13	@KerysHoran @snapchatsupport I went to same issue last week, my Instagram was hacked but #cyberking	1.55E+18	1.55E+18			snapchat
14	@snapchatsupport can NOT access front camera on galaxy s22 Ultra. When I try to "flip" to front camera it	210335751	1.55E+18			snapchat
15	@snapchatsupport fix it omg it,Äôs been a fuckimg day	1.29E+18	1.55E+18	sxdek	United Kingdom	snapchat
16	@alexolix05 @snapchatsupport Moi aussi, je suis dV©connectV©e depuis le matin ca saouuuule	1.51E+18	1.55E+18			snapchat
17	@boredowntw @snapchatsupport yea i just contacted customer support on there website. idk if its a malfu	1.34E+18	1.55E+18	PENIELMARAJ	□üá¨□üá≠ || □ùêõ□ùê•□ùê§ || □ùê°	snapchat
18	@alexolix05 @lostsoulnw @snapchatsupport T,Äôas de la chance, moi impossible de ne serait-ce me conn	1.54E+18	1.55E+18			snapchat
19	@sosaaaaa57 @snapchatsupport I went to same issue last week, my Instagram was hacked but #cyberking	1.55E+18	1.55E+18			snapchat
20	@BTS_twttsss @snapchatsupport You won,Äôt get help so move on ,úÖ	1.55E+18	1.55E+18			snapchat
21	@snapchatsupport @Snapchat	1499319673	1.55E+18	JacobBlackey	Belmont, NH	snapchat
22	@snapchatsupport when switching between cameras on my s22, i cant switch to frontfacing camera, but it	1.09E+18	1.55E+18	DrMagnusW1	Steinkjer, Norge	snapchat
23	@snapchatsupport Mine still ain,Äôt working and y,Äôall couldn,Äôt help me via DM nor email,Ä¶ □ü§¶□üè	1.12E+18	1.55E+18	lexiLeighaHart	Florida, USA	snapchat
24	@KerysHoran @snapchatsupport I went to same issue last week, my Instagram was hacked but #cyberking	1.55E+18	1.55E+18			snapchat
25	@snapchatsupport can NOT access front camera on galaxy s22 Ultra. When I try to "flip" to front camera it	210335751	1.55E+18			snapchat
26	@snapchatsupport fix it omg it,Äôs been a fuckimg day	1.29E+18	1.55E+18	sxdek	United Kingdom	snapchat
27	@snapchatsupport I can,Äôt login to my Snapchat account it,Äôs been like this for 2 days now. I have dm,Ä	1.52E+18	1.55E+18			snapchat
28	@sean_deeks @Snapchat @snapchatsupport Shit passing me off.	8.48E+17	1.55E+18	TyDelbridge	Oregon, USA	snapchat
29	Hello??!!! @snapchatsupport @Snapchat @SnapforDevs I,Äôve been tweeting you and no response for like	1.28E+18	1.55E+18			snapchat
30	@aden_landy @JohnMon20597405 @snapchatsupport @oGHisses Mine just keeps zooming in out but wor	1.45E+18	1.55E+18			snapchat
31	@abyarxx27 @snapchatsupport same dude	1.39E+18	1.55E+18			snapchat
32	@JohnMon20597405 @snapchatsupport @oGHisses Mine too	9.07E+17	1.55E+18			snapchat
33	@VictoriaChay4 @snapchatsupport It was frustrating when I lost access to my account and I lost hope in ge	1.55E+18	1.55E+18			snapchat
34	@gamermika19 @snapchatsupport Same here	1.47E+18	1.55E+18			snapchat
35	@sxdek @snapchatsupport inittt omg	1.47E+18	1.55E+18			snapchat
36	@Deeppatel145 @snapchatsupport Chat up now for quick recovery of your account	1.49E+18	1.55E+18			snapchat
37	@lilkrevived @snapchatsupport @Snapchat Chat up now for quick recovery of your account	1.49E+18	1.55E+18			snapchat
38	@snapchatsupport Just sent a DM.	280254902	1.55E+18	lenessakeehn	Deadwood, SD	snapchat

4.3 Implementation

Installation

To do sentimental analysis, we have been using Google Colab notebook. Installing the required libraries, such as the VADER library for the VADER sentimental analysis and the pinguoin library from the scipy package for the Mann-Whitney U test, is a must.

```
!pip install vaderSentiment
!pip install scipy pandas pingouin
```

Figure 15 Installing libraries

Importing necessary libraries

For evaluation, we have taken six software product support accounts: Snapchat, Instagram, Facebook, Messenger, WhatsApp, and Telegram.

All the needed library files must be imported. Pandas, NumPy, and Matplotlib, are just a few Python packages required for data collection and preparation. Using the vaderSentiment API, we imported the required package SentimentIntensityAnalyzer to make the object.

We have successfully retrieved 15024 tweets from the software support account on Twitter that include geolocation information.

Upload the dataset.csv file created using script 2 to the Google Collaboratory, and then convert and save it using the pandas library in a data frame.

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pycountry
import math
from scipy.stats import mannwhitneyu
```

```
[ ] filepath = '/content/dataset.csv'
```

```
[ ] dataframe = pd.read_csv(filepath,on_bad_lines='skip')
```

```
[ ] dataframe.head()
```

<table>
<tr><th></th><th>Text</th><th>Author_id</th><th>Id</th><th>Username</th><th>Location</th><th>Account</th></tr>
<tr><td>0</td><td>@boredowntw @snapchatsupport yea i just contac...</td><td>1.340000e+18</td><td>1.550000e+18</td><td>PENIELMARAJ</td><td>🇬🇭 || blk || he || bi</td><td>snapchat</td></tr>
<tr><td>1</td><td>@alexolix05 @lostsoulnw @snapchatsupport T'as ...</td><td>1.540000e+18</td><td>1.550000e+18</td><td>NaN</td><td>NaN</td><td>snapchat</td></tr>
<tr><td>2</td><td>@snapchatsupport I stil can't log in to my acc...</td><td>1.400000e+18</td><td>1.550000e+18</td><td>gamermika19</td><td>Roermond, Nederland</td><td>snapchat</td></tr>
<tr><td>3</td><td>@snapchatsupport @snapchatsupport @Snapchat I...</td><td>1.550000e+18</td><td>1.550000e+18</td><td>NaN</td><td>NaN</td><td>snapchat</td></tr>
<tr><td>4</td><td>@snapchatsupport Help i have been having this ...</td><td>1.470000e+18</td><td>1.550000e+18</td><td>NaN</td><td>NaN</td><td>snapchat</td></tr>
</table>

Figure 16 Importing libraries

Using the constructor for the SentimentIntensityAnalyzer class, a new VADER sentimental analyzer object with the name "analyser" is created ().

```
[ ] analyser = SentimentIntensityAnalyzer()
```

Figure 17 Cleaning the Twitter data

We are cleaning the tweets by writing two methods, remove_pattern() and clean_tweets(), respectively.

Now that we have collected real-time tweets from Twitter, we must process the information. It requires cleaning data, which is a challenge when analyzing tweets due to the prevalence of URLs, numbers, and user ids.

```
[ ] #cleaning the tweets
    def remove_pattern(input_txt, pattern):
        r = re.findall(pattern, input_txt)
        for i in r:
            input_txt = re.sub(i, '', input_txt)
        return input_txt
    def clean_tweets(tweets):
        #remove twitter Return handles (RT @xxx:)
        tweets = np.vectorize(remove_pattern)(tweets, "RT @[\w]*:")

        #remove twitter handles (@xxx)
        tweets = np.vectorize(remove_pattern)(tweets, "@[\w]*")

        #remove URL links (httpxxx)
        tweets = np.vectorize(remove_pattern)(tweets, "https?://[A-Za-z0-9./]*")

        #remove special characters, numbers, punctuations (except for #)
        tweets = np.core.defchararray.replace(tweets, "[^a-zA-Z]", " ")

        return tweets
```

Figure 18 Clean tweet logic

clean_tweets() takes tweet text content as input. Hence, the df['text'] full column is now referred to as the clean tweets() function argument.

```
[ ] dataframe['Text'] = clean_tweets(dataframe['Text'])
    dataframe['Text'].head()

    0        yea i just contacted customer support on the...
    1         T’as de la chance, moi impossible de ne ser...
    2       I stil can’t log in to my account it says bc ...
    3          I have not been received an email or anyth...
    4       Help i have been having this issue since yest...
    Name: Text, dtype: object
```

Figure 19 Displaying tweet content

Finding scores for tweets

We are creating four lists: positive, negative, neutral, and compound. These lists save the respective scores generated by invoking the VADER analyzer method known as "analyser.polarity scores()," bypassing the tweet text as a function argument from the data frame iterated using them for a loop. These lists will be used to save the respective scores.

Once all scores have been obtained, a dictionary is constructed and added to the list, simply called "scores."

```
[ ] scores = []
    # Declare variables for scores
    compound_list = []
    positive_list = []
    negative_list = []
    neutral_list = []
    for i in range(dataframe['Text'].shape[0]):
    #print(analyser.polarity_scores(sentiments_pd['text'][i]))
        compound = analyser.polarity_scores(dataframe['Text'][i])["compound"]
        pos = analyser.polarity_scores(dataframe['Text'][i])["pos"]
        neu = analyser.polarity_scores(dataframe['Text'][i])["neu"]
        neg = analyser.polarity_scores(dataframe['Text'][i])["neg"]

        scores.append({"Compound": compound,
                           "Positive": pos,
                           "Negative": neg,
                           "Neutral": neu
                      })
```

Figure 20 Tweet score analyser logic

Join the sentiments score data frame with the df data frame by first converting the scores dictionary holding the scores into the data frame. We may now examine the tweet's positivity, neutrality, or negativity based on its score.

```
[ ] sentiments_score = pd.DataFrame.from_dict(scores)
    dataframe = dataframe.join(sentiments_score)
    dataframe.head()
```

	Text	Author_id	Id	Username	Location	Account	Compound	Positive	Negative	Neutral
0	yea i just contacted customer support on the...	1.340000e+18	1.550000e+18	PENIELMARAJ	🇬🇭 ll blk ll he ll bi	snapchat	-0.6597	0.077	0.259	0.664
1	T'as de la chance, moi impossible de ne ser...	1.540000e+18	1.550000e+18	NaN	NaN	snapchat	0.2500	0.167	0.000	0.833
2	I stil can't log in to my account it says bc ...	1.400000e+18	1.550000e+18	gamermika19	Roermond, Nederland	snapchat	0.1855	0.101	0.109	0.790
3	I have not been received an email or anyth...	1.550000e+18	1.550000e+18	NaN	NaN	snapchat	0.4019	0.130	0.000	0.870
4	Help i have been having this issue since yest...	1.470000e+18	1.550000e+18	NaN	NaN	snapchat	0.5267	0.157	0.067	0.776

Figure 21 Data frame with positive, negative, and neutral scores

The total score is a normalized scale from -1 (most severe negative) to +1 (most extreme positive) based on the sum of the positive, negative, and neutral weights (most extreme positive).

The closer a Compound score is to +1, the more positively biased the text is.

We provide a positive or negative label to the tweet content based on the compound score values of the VADER; a function named **sentimentPredict()** was built, and the condition shown below has been implemented into the function. The required labels have been appended to the tweet rows in the CSV file.

positive sentiment : (compound score >= 0.05)

neutral sentiment : (compound score > -0.05) and (compound score < 0.05)

negative sentiment : (compound score <= -0.05)

```
def sentimentPredict(sentiment):
  if sentiment >= 0.05:
    return "Positive"
  elif sentiment <= -0.05:
    return "Negative"
  else:
    return "Neutral"
dataframe['label'] =dataframe['Compound'].apply(lambda x: sentimentPredict(x))
dataframe.head(10)
```

	Text	Author_id	Id	Username	Location	Account	Compound	Positive	Negative	Neutral	label
0	yea i just contacted customer support on the...	1.340000e+18	1.550000e+18	PENIELMARAJ	🇬🇭 || blk || he || bi	snapchat	-0.6597	0.077	0.259	0.664	Negative
1	T'as de la chance, moi impossible de ne ser...	1.540000e+18	1.550000e+18	NaN	NaN	snapchat	0.2500	0.167	0.000	0.833	Positive
2	I stil can't log in to my account it says bc ...	1.400000e+18	1.550000e+18	gamermika19	Roermond, Nederland	snapchat	0.1855	0.101	0.109	0.790	Positive
3	I have not been received an email or anyth...	1.550000e+18	1.550000e+18	NaN	NaN	snapchat	0.4019	0.130	0.000	0.870	Positive
4	Help i have been having this issue since yest...	1.470000e+18	1.550000e+18	NaN	NaN	snapchat	0.5267	0.157	0.067	0.776	Positive
5	Hello??!!! I've been tweeting you and no re...	1.430000e+18	1.550000e+18	NaN	NaN	snapchat	0.6836	0.223	0.109	0.668	Positive
6	how long is temporary	1.480000e+18	1.550000e+18	NaN	NaN	snapchat	0.0000	0.000	0.000	1.000	Neutral
7	Exact problem I'm having now and yes earli...	3.449564e+08	1.550000e+18	NaN	NaN	snapchat	0.2023	0.296	0.178	0.526	Positive
8	Mine won't work either and I ahe just S22	1.450000e+18	1.550000e+18	NaN	NaN	snapchat	0.0000	0.000	0.000	1.000	Neutral
9	when switching between cameras on my s22, i c...	1.090000e+18	1.550000e+18	DrMagnusW1	Steinkjer, Norge	snapchat	0.5499	0.139	0.000	0.861	Positive

Figure 22 Label output in the data frame for the tweet text

The image shows positive, negative, and neutral word counts.

```
import seaborn as sns
sns.countplot(dataframe['label'], label='count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f975dcb43d0>
```

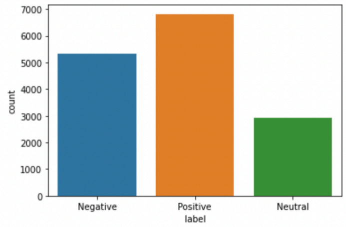

Figure 23 positive, negative and neutral count value

Plotting word cloud

Using a word cloud, We can see which terms appear more frequently in our text or data collection. Wordcloud is a useful visual representation of text information, showcasing the most frequently occurring and consequential terms in text datasets.

Put up a word cloud of all the terms used in the tweets.

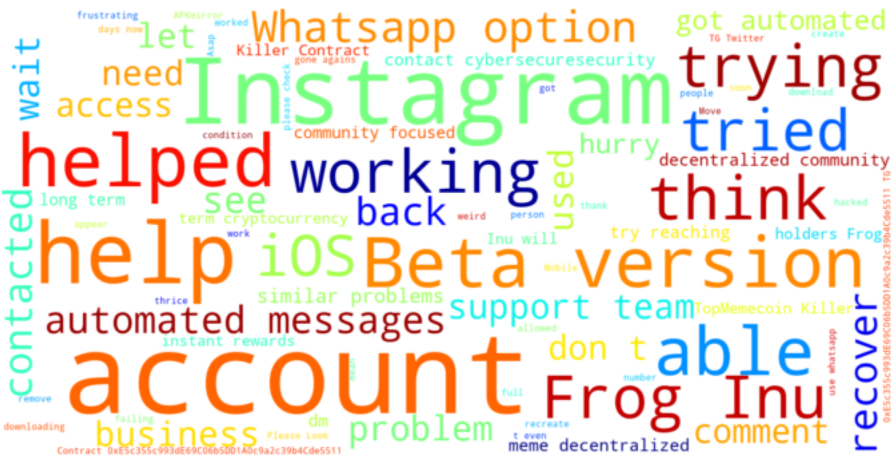

Figure 24 Word cloud of entire tweet content

From the above word cloud **Instagram, account, Beta, and help** are the words that appear most frequently time in the tweets dataset.

The following logic is for displaying the negative words that are present in the dataset fetched from Twitter

```
neg_sentences = negative['Text'].tolist()
neg_sentences_as_one_string = " ".join(neg_sentences)
plt.figure(figsize = (15,15))
wordcloud = WordCloud(
        background_color='white',
        width=1600,
        height=800,
        random_state=1,
        colormap='jet',
        max_words=80,
        max_font_size=200).generate(neg_sentences_as_one_string)
plt.figure(figsize=(12, 10))
plt.axis('off')
plt.imshow(wordcloud, interpolation="bilinear");
plt.imshow(wordcloud)
```

```
<matplotlib.image.AxesImage at 0x7f975ad72e90>
<Figure size 1080x1080 with 0 Axes>
```

Figure 25 Negative Word Cloud

The following logic is for displaying the positive words that are present in the dataset fetched from Twitter

```
pos_sentences = positive['Text'].tolist()
pos_sentences_as_one_string = " ".join(pos_sentences)
plt.figure(figsize = (15,15))
wordcloud = WordCloud(
        background_color='white',
        width=1600,
        height=800,
        random_state=1,
        colormap='jet',
        max_words=80,
        max_font_size=200).generate(pos_sentences_as_one_string)
plt.figure(figsize=(12, 10))
plt.axis('off')
plt.imshow(wordcloud, interpolation="bilinear");
plt.imshow(wordcloud)
```

```
<matplotlib.image.AxesImage at 0x7f975ac163d0>
<Figure size 1080x1080 with 0 Axes>
```

Figure 26 Positive Word Cloud

The following logic is for displaying the neutral words that are present in the dataset fetched from Twitter

```
neu_sentences = neutral['Text'].tolist()
neu_sentences_as_one_string = " ".join(neu_sentences)
plt.figure(figsize = (15,15))
wordcloud = WordCloud(
        background_color='white',
        width=1600,
        height=800,
        random_state=1,
        colormap='jet',
        max_words=80,
        max_font_size=200).generate(neu_sentences_as_one_string)
plt.figure(figsize=(12, 10))
plt.axis('off')
plt.imshow(wordcloud, interpolation="bilinear");
plt.imshow(wordcloud)
```

```
<matplotlib.image.AxesImage at 0x7f975cf63f10>
<Figure size 1080x1080 with 0 Axes>
```

Figure 27 Neutral Word Cloud

An evaluation of the average compound score of all tweets from the software product support account is presented as a pivot table, which is also displayed.

```
#Collect the compound values for each news source
score_table = dataframe.pivot_table(index='Account',  values="Compound", aggfunc = np.mean)
score_table
```

Account	Compound
facebook	-0.031306
instagram	0.081103
messenger	-0.072470
snapchat	0.077882
telegram	0.164028
whatsapp	0.051440

Figure 28 Compound values pivot table

In the above image, it can be inferred that the compound score of the **Telegram** software product is the highest, and the minus symbol in the compound score indicates that it possesses negative tweets.

Now looking for the software product support accounts that posted a maximum number of negative tweets

```
[ ] #Collect the negative values for each news source
    neg_score_table = dataframe.pivot_table(index='Account',  values="Negative", aggfunc = np.mean)
    neg_score_table
```

Account	Negative
facebook	0.140315
instagram	0.088443
messenger	0.123800
snapchat	0.091146
telegram	0.099500
whatsapp	0.054700

Figure 29 Negative values pivot table

The Polarity score has now been revealed. Representation of the pivot table for negative scores is as follows:

```
[ ] neg_score_table.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f975b204090>
```

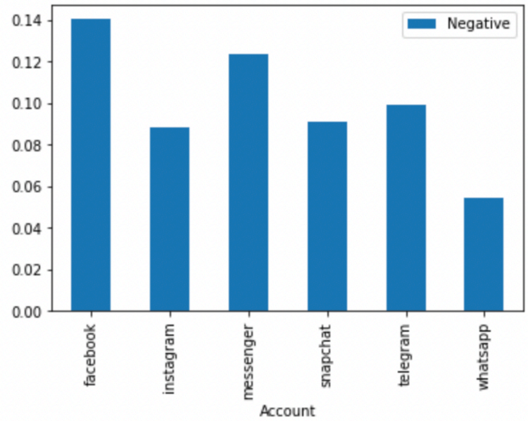

Figure 30Account vs. Software Product Plot(Negative score)

The Polarity score has now been revealed. Representation of the pivot table for positive scores is as follows:

```
pos_score_table = dataframe.pivot_table(index='Account',  values="Positive", aggfunc = np.mean)
pos_score_table
```

Account	Positive
facebook	0.119925
instagram	0.128725
messenger	0.083400
snapchat	0.097439
telegram	0.162247
whatsapp	0.071500

```
[ ] pos_score_table.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f975abf9fd0>
```

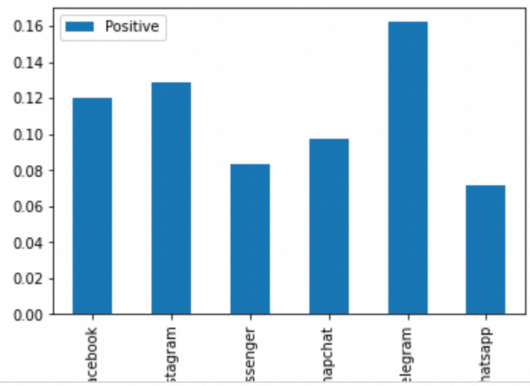

Figure 31Account vs. Software Product Plot(Positive score)

Filter out the tweet rows with a location value that is not null.

```
loca_df = dataframe[dataframe['Location'].notnull()]
len(loca_df)
```

```
9149
```

```
[ ] loca_df
```

	Text	Author_id	Id	Username	Location	Account	Compound	Positive	Negative	Neutral	label
0	yea i just contacted customer support on the...	1.340000e+18	1.550000e+18	PENIELMARAJ	🇬🇭 || **blk** || **he** || **bi**	snapchat	-0.6597	0.077	0.259	0.664	Negative
2	I stil can't log in to my account it says bc ...	1.400000e+18	1.550000e+18	gamermika19	Roermond, Nederland	snapchat	0.1855	0.101	0.109	0.790	Positive
9	when switching between cameras on my s22, i c...	1.090000e+18	1.550000e+18	DrMagnusW1	Steinkjer, Norge	snapchat	0.5499	0.139	0.000	0.861	Positive
10	Mine still ain't working and y'all couldn't h...	1.120000e+18	1.550000e+18	lexiLeighaHart	Florida, USA	snapchat	0.4019	0.105	0.000	0.895	Positive
13	fix it omg it's been a fuckimg day	1.290000e+18	1.550000e+18	sxdek	United Kingdom	snapchat	0.0000	0.000	0.000	1.000	Neutral
...	...	...	...	...	...	...	...	...	...	...	...
15065	So Facebook disabled my account for some rando...	1.197863e+08	1.560000e+18	TCapitalG	CincinnatiL.A.lCorpus Christi	facebook	0.0000	0.000	0.000	1.000	Neutral
15067	And you will pay the price one day for lyi...	7.732281e+08	1.560000e+18	HAdamsen	Hillerød, Danmark	facebook	-0.7351	0.000	0.298	0.702	Negative
15068	is there someway to contact you guys directly...	1.023073e+08	1.560000e+18	OdrarEth	Florida, USA	facebook	-0.5994	0.000	0.123	0.877	Negative
15070	😆\n\nI'll post them as soon as my ban is ove...	7.113803e+08	1.560000e+18	lifelong_lfc	North West, England	facebook	-0.3382	0.117	0.182	0.701	Negative
15071	So Facebook disabled my account for some rando...	1.197863e+08	1.560000e+18	TCapitalG	CincinnatiL.A.lCorpus Christi	facebook	0.0000	0.000	0.000	1.000	Neutral

Figure 32 Location filtered data frame output

Split the location value into city and country columns in the data frame. The resultant data frame will look like the below:

9149 rows × 11 columns

```
temp_df = loca_df['Location'].str.split(',', expand=True)
temp_df.columns= ['city','country']
new_loc_df = pd.concat([loca_df,temp_df], axis=1,ignore_index=False)
new_loc_df
```

	Text	Author_id	Id	Username	Location	Account	Compound	Positive	Negative	Neutral	label	city	country
0	yea i just contacted customer support on the...	1.340000e+18	1.550000e+18	PENIELMARAJ	🇬🇭 || **blk** || **he** || **bi**	snapchat	-0.6597	0.077	0.259	0.664	Negative	🇬🇭 || **blk** || **he** || **bi**	None
2	I stil can't log in to my account it says bc ...	1.400000e+18	1.550000e+18	gamermika19	Roermond, Nederland	snapchat	0.1855	0.101	0.109	0.790	Positive	Roermond	Nederland
9	when switching between cameras on my s22, i c...	1.090000e+18	1.550000e+18	DrMagnusW1	Steinkjer, Norge	snapchat	0.5499	0.139	0.000	0.861	Positive	Steinkjer	Norge
10	Mine still ain't working and y'all couldn't h...	1.120000e+18	1.550000e+18	lexiLeighaHart	Florida, USA	snapchat	0.4019	0.105	0.000	0.895	Positive	Florida	USA
13	fix it omg it's been a fuckimg day	1.290000e+18	1.550000e+18	sxdek	United Kingdom	snapchat	0.0000	0.000	0.000	1.000	Neutral	United Kingdom	None
...	...	...	...	...	...	...	...	...	...	...	...	...	...
	So												

Figure 33 Split location into city and country

After segregating into city and country columns, filter out the rows in the data frame with country values, not null.

```
new_loc_df = new_loc_df[new_loc_df['country'].notnull()]
```

```
new_loc_df
```

	Text	Author_id	Id	Username	Location	Account	Compound	Positive	Negative	Neutral	label	city	country
2	I stil can't log in to my account it says bc ...	1.400000e+18	1.550000e+18	gamermika19	Roermond, Nederland	snapchat	0.1855	0.101	0.109	0.790	Positive	Roermond	Nederland
9	when switching between cameras on my s22, i c...	1.090000e+18	1.550000e+18	DrMagnusW1	Steinkjer, Norge	snapchat	0.5499	0.139	0.000	0.861	Positive	Steinkjer	Norge
10	Mine still ain't working and y'all couldn't h...	1.120000e+18	1.550000e+18	lexiLeighaHart	Florida, USA	snapchat	0.4019	0.105	0.000	0.895	Positive	Florida	USA
19		1.499320e+09	1.550000e+18	JacobBlackey	Belmont, NH	snapchat	0.0000	0.000	0.000	0.000	Neutral	Belmont	NH
20	when switching between cameras on my s22, i c...	1.090000e+18	1.550000e+18	DrMagnusW1	Steinkjer, Norge	snapchat	0.5499	0.139	0.000	0.861	Positive	Steinkjer	Norge
...	...	...	...	...	...	...	...	...	...	...	...	...	...
15062	is there someway to contact you guys directly...	1.023073e+08	1.560000e+18	OdrarEth	Florida, USA	facebook	-0.5994	0.000	0.123	0.877	Negative	Florida	USA
15064	😔\n\nI'll post them as soon as my ban is ove...	7.113803e+08	1.560000e+18	lifelong_lfc	North West, England	facebook	-0.3382	0.117	0.182	0.701	Negative	North West	England
15067	And you will pay the price one day for lyi...	7.732281e+08	1.560000e+18	HAdamsen	Hillerød, Danmark	facebook	-0.7351	0.000	0.298	0.702	Negative	Hillerød	Danmark
15068	is there someway to contact you guys directly...	1.023073e+08	1.560000e+18	OdrarEth	Florida, USA	facebook	-0.5994	0.000	0.123	0.877	Negative	Florida	USA
15070	😔\n\nI'll post them as soon as my ban is ove...	7.113803e+08	1.560000e+18	lifelong_lfc	North West, England	facebook	-0.3382	0.117	0.182	0.701	Negative	North West	England

4617 rows × 13 columns

Figure 34 Filtering, not null country rows

Create a pivot table at the country level and organize all the software product support accounts under it with corresponding positive scores.

```
loc_based_pos_score_table = new_loc_df.pivot_table(index=['country','Account'], values="Positive", aggfunc = np.mean)
```

```
loc_based_pos_score_table
```

country	Account	Positive
Austria	messenger	0.078000
CA	facebook	0.289000
	whatsapp	0.000000
Danmark	facebook	0.000000
England	facebook	0.117000
	instagram	0.201000
	whatsapp	0.000000
GA	facebook	0.380493
IL	snapchat	0.000000
IN	instagram	0.000000
India	instagram	0.270298
	snapchat	0.435000
	telegram	0.113000
	whatsapp	0.027333
MD	snapchat	0.261500
MO	snapchat	0.098000

Figure 35 Country-level Software Account filtering of positive score

Plotting for the country-level software support account with positive scores

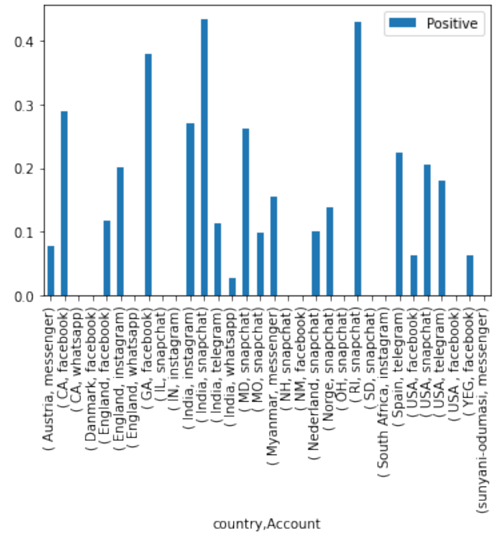

Figure 36Plotting for country-level software account(positive)

Create a pivot table at the country level and organize all the software product support accounts under it with corresponding negative scores.

```
loc_based_neg_score_table = new_loc_df.pivot_table(index=['country','Account'], values="Negative", aggfunc = np.mean)
```

```
[ ] loc_based_neg_score_table
```

country	Account	Negative
Austria	messenger	0.093000
CA	facebook	0.447000
	whatsapp	0.000000
Danmark	facebook	0.298000
England	facebook	0.182000
	instagram	0.075000
	whatsapp	0.088000
GA	facebook	0.213178
IL	snapchat	0.508000
IN	instagram	0.000000
India	instagram	0.141196
	snapchat	0.116000
	telegram	0.097000
	whatsapp	0.113333
MD	snapchat	0.000000
MO	snapchat	0.156000

Figure 37 Country-level Software Account filtering of negative score

Negative scores for country-level software support

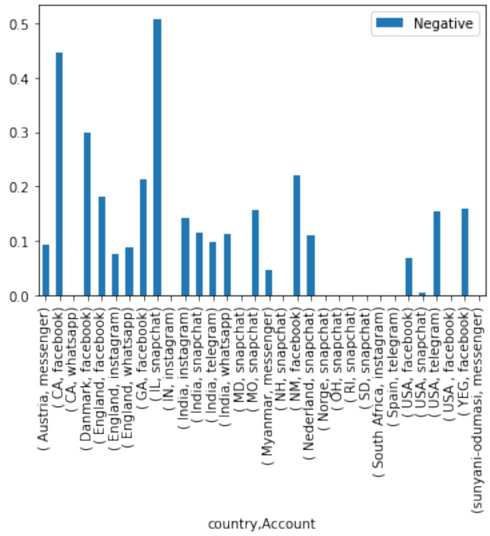

Figure 38 Plotting for country-level software account(negative)

Create a pivot table at the country level and organize all the software product support accounts under it with corresponding compound scores.

Mann Whitney U test

We are applying the Mann-Whitney U test on the compound score pairs at the location level and generating the p-value for it.

In the below figure, we are filtering out the data frame based on the country name "England" and account value "Instagram." After that, fetch the compound values of the data frame and create a list.

```
#l1 = new_loc_df[[' England',  'instagram']]

df_tp1 = new_loc_df[(new_loc_df["country"] == ' England') & (new_loc_df["Account"] == 'instagram')]
df_tp2 = new_loc_df[(new_loc_df["country"] == ' England') & (new_loc_df["Account"] == 'whatsapp')]
l1 = df_tp1['Compound'].tolist()
l2 = df_tp2['Compound'].tolist()
final_list1 = l1+l2

df_tp3 = new_loc_df[(new_loc_df["country"] == ' India') & (new_loc_df["Account"] == 'instagram')]
df_tp4 = new_loc_df[(new_loc_df["country"] == ' India') & (new_loc_df["Account"] == 'whatsapp')]
l3 = df_tp3['Compound'].tolist()
l4 = df_tp4['Compound'].tolist()
final_list2 = l3 + l4
```

```
[89] len(final_list1)

450
```

```
[90] len(final_list2)

1576
```

Figure 39 data frame generation of Software Account and Country pairs

Construct a data frame out of compound values of Instagram and WhatsApp software product accounts for England and India.

```
[98] data_dict = {'Eng':final_list1,"Ind":final_list2}

     df = pd.DataFrame({'Eng': pd.Series(final_list1), 'Ind': pd.Series(final_list2)})
     df = df.replace(np.nan, 0)
     df
```

We passed the generated pair for the Mann-Whitney U test. As a result, statistic and p-value get generated.

```
[99] results = mannwhitneyu(df['Eng'], df['Ind'])
     results

     MannwhitneyuResult(statistic=1429139.0, pvalue=3.1077084834410555e-15)
```

Here we generate data frames by filtering out compound scores at the software product support account level. Only two accounts, Facebook and Instagram, were considered here, and lists are generated for the same.

```
[ ] df_tp1 = new_loc_df[(new_loc_df["country"] == ' Danmark') & (new_loc_df["Account"] == 'facebook')]
    df_tp2 = new_loc_df[(new_loc_df["country"] == ' England') & (new_loc_df["Account"] == 'facebook')]
    df_tp3 = new_loc_df[(new_loc_df["country"] == ' USA') & (new_loc_df["Account"] == 'facebook')]
    l1 = df_tp1['Compound'].tolist()
    l2 = df_tp2['Compound'].tolist()
    l3 = df_tp3['Compound'].tolist()
    final_list1 = l1+l2

    df_tp4 = new_loc_df[(new_loc_df["country"] == ' England') & (new_loc_df["Account"] == 'instagram')]
    df_tp5 = new_loc_df[(new_loc_df["country"] == ' India') & (new_loc_df["Account"] == 'instagram')]
    df_tp6 = new_loc_df[(new_loc_df["country"] == ' South Africa') & (new_loc_df["Account"] == 'instagram')]
    l4 = df_tp4['Compound'].tolist()
    l5 = df_tp5['Compound'].tolist()
    l6 = df_tp6['Compound'].tolist()
    final_list2 = l3 + l4 +l6



    data_dict = {'Fb':final_list1,"Inst":final_list2}

    df_fb_inst = pd.DataFrame({'Fb': pd.Series(final_list1), 'Inst': pd.Series(final_list2)})
    df_fb_inst = df_fb_inst.replace(np.nan, 0)
```

Figure 40data frame generation of Software Account pairs

These list values are passed to the data frame with Facebook and Instagram indices. The constructed pairs were passed to the test, and the p-value was generated using it.

```
[ ]  results = mannwhitneyu(df_fb_inst['Fb'], df_fb_inst['Inst'])
     results

     MannwhitneyuResult(statistic=140795.0, pvalue=4.17604243596908e-21)
```

The Mann-Whitney U test pairs were applied to the tweet dataset of the various software support account pairs: Facebook & Instagram, Snapchat & Instagram, Facebook & Whatsapp, Instagram & Whatsapp, and Facebook & Snapchat. Below are the screenshots for the pairs mentioned above.

Facebook & Instagram

```
[66]  results = mannwhitneyu(df_fb_inst['Fb'], df_fb_inst['Inst'])
      results

      MannwhitneyuResult(statistic=143144.5, pvalue=0.036003201427796325)
```

Snapchat & Instagram

```
[68]  results = mannwhitneyu(df_fb_inst['Sn'], df_fb_inst['Ins'])
      results

      MannwhitneyuResult(statistic=88320.0, pvalue=3.933134314972951e-06)
```

Whatsapp & Snapchat

```
[70]  results = mannwhitneyu(df_fb_inst['Wh'], df_fb_inst['Sn'])
      results

      MannwhitneyuResult(statistic=0.0, pvalue=1.3311138118554098e-14)
```

Facebook & Whatsapp

```
[72] results = mannwhitneyu(df_fb_inst['Fb'], df_fb_inst['What'])
     results

     MannwhitneyuResult(statistic=162892.5, pvalue=0.06173878799528974)
```

Instagram & Whatsapp

```
results = mannwhitneyu(df_fb_inst['Ins'], df_fb_inst['What'])
results

MannwhitneyuResult(statistic=66690.0, pvalue=0.00024544970366060256)
```

Facebook & Snapchat

```
results = mannwhitneyu(df_fb_inst['Fb'], df_fb_inst['Sn'])
results

MannwhitneyuResult(statistic=159894.0, pvalue=0.22390255208847598)
```

5. Discussion and Results:

VADER does a great job at organizing the feelings. Whether looking to analyze reviews or social media posts, this ready-made model may be utilized with little to no modification. The icing is that VADER needs zero training data to function. The Mann-Whitney U test compares the two groups when the dependent variable is ordinal or continuous but not normally distributed. This test has been conducted on the pairs of the dataset, which infers that it is not normally distributed.

The below table shows the resultant p-value of the Mann-Whitney U test of the software product account pairs.

Mann Whitney U test pairs	Facebook, Instagram	Snapchat, Instagram	Whatsapp, Snapchat	Facebook, Whatsapp	Instagram, Whatsapp	Facebook, Snapchat
P value	0.03600	0.0000039	0.0000000000000133	0.06173	0.000245	0.2239

Table 1Mann Whitney U test P value scores

Mann Whitney U test P value scores

The p-value of all the pairs has less than 0.05, which shows that the tweets in the dataset are distinctly significant to each other, excluding Facebook, Whatsapp and Facebook, and Snapchat pairs. It is because those two pairs have values greater than 0.05, which shows that the dataset is insignificant to each other.

6. Conclusion

In this study, we analyzed user comments on software from the Twitter Support Accounts of six applications in different nations to determine if there were any discernible changes in the dataset between countries. Requirements engineers and other stakeholders inside software firms may use the data from the Mann-Whitney U test to determine how critical the tweet text content is to each software combination. It is fascinating to think about performing sentiment analysis on social media text data with VADER. Understanding customers' feelings about a company's goods and services are crucial to maintaining a solid brand identity. Thus, doing regular sentiment research is a must.

7. Future Work

Feedback from users is essential for making software more error- and need-free in the future. Processing algorithms must take into account user feedback from a wide range of people on Twitter about software as part of their development, testing, and validation phases in order to reduce the likelihood of any underlying cultural biases and to increase understanding of why it is essential to analyze feedback from a wide range of people. In the future, this analyzed stream of tweets may be used to categorize bug reports, feature requests, and help inquiries for each software product's support accounts worldwide—this aids in the development and improvement of the program.

8. References:

Qualtrics Blog. (2022, August 20). *Qualtrics Sentimental Analysis*. Retrieved from Qualtrics Sentimental Analysis: https://www.qualtrics.com/uk/experience-management/research/sentiment-analysis/

Palakorn Achananuparp, I. N.-P. (2011). Observatory of Trends in Software Related Microblogs. *IEEE/ACM International Conference*, 1-2.

Walaa Medhat, A. H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 1-3.

Maleknaz Nayebi, H. C. (2018). App store mining is not enough for app improvement. *Empir Software Eng*, 1-2.

Daniel Martens, W. M. (2019). Extracting and Analyzing Context Information in User-Support Conversations on Twitter. *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 1-2.

Isabel Anger, C. K. (2011). Measuring Influence on Twitter . *i-KNOW '11: Proceedings of the 11th International Conference* , 2.

Lyon, E. (2011, October 1). *Differing Approaches to Social Influence*. Retrieved from Sparxoo: https://sparxoo.com/2009/10/01/differing-approaches-to-social-influence/

Bucher, T. (2013, November 16). *Objects of Intense Feeling: The Case of the Twitter API*. Retrieved from Computational Culture a journal of software studies: http://computationalculture.net/objects-of-intense-feeling-the-case-of-the-twitter-api/

Alina Campan, T. A. (2018). Is Data Collection through Twitter Streaming API Useful for Academic Research? . *2018 IEEE International Conference on Big Data (Big Data)*, 1-2.

Roesslein, J. (2018). tweepy Documentation. *https://buildmedia.readthedocs.org/media/pdf/tweepy/v3.6.0/tweepy.pdf*, 17. Retrieved from https://buildmedia.readthedocs.org/

Garcia, M. (2022, August 21). *How to Make a Twitter Bot in Python With Tweepy*. Retrieved from Real Python: https://realpython.com/twitter-bot-python-tweepy/

Twitter. (2022, August 21). *About the Twitter API*. Retrieved from Twitter API access levels and versions: https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api

Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. *Communications of the ACM*, 1.

Mejova, Y. (2009). Sentiment Analysis: An Overview. *Academia*, 5.

Calderon, P. (2017, April 10). *VADER Sentiment Analysis Explained*. Retrieved from medium: https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd9

Swarnkar, N. (2020, May 21). *VADER Sentiment Analysis: A Complete Guide, Algo Trading and More*. Retrieved from blog.quantinsti: https://blog.quantinsti.com/vader-sentiment/

Kung, S. (2021, May 1). *Social Media Sentiment Analysis with VADER*. Retrieved from towardsdatascience: https://towardsdatascience.com/social-media-sentiment-analysis-with-vader-c29d0c96fa90

White, B. (2020, May 27). *Sentiment Analysis: VADER or TextBlob*. Retrieved from towardsdatascience: https://towardsdatascience.com/sentiment-analysis-vader-or-textblob-ff25514ac540

C.J. Hutto, E. G. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text . *AAAI Conference* , 1-2.

Marsja, E. (2020, September 26). *How to Perform Mann-Whitney U Test in Python with Scipy and Pingouin*. Retrieved from marsja.se: https://www.marsja.se/how-to-perform-mann-whitney-u-test-in-python-with-scipy-and-pingouin/

Karadimitriou, S. M. (2018). Mann-Whitney U test (Nonparametric equivalent to independent samples t-test). *Sheffield: Sheffield Hallam University.*, 1.

Nachar, N. (2008). The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution. *Tutorials in Quantitative Methods for Psychology*, 1-2.

Emitza Guzman, R. A. (2016). A Needle in a Haystack: What Do Twitter Users Say about Software? *IEEE 24th International Requirements Engineering Conference*, 1.

Soo Ling Lim, P. J. (2015). Investigating Country Differences in Mobile App User Behavior and Challenges for Software Engineering. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 1.

Maleknaz Nayebi, H. C. (2018). App store mining is not enough for app improvement. *Empir Software Eng* , 1.

Emitza Guzman, M. I. (2017). Prioritizing User Feedback from Twitter: A Survey Report. *IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering* , 1.

Grant Williams, A. M. (2017). Mining Twitter Feeds for Software User Requirements. *IEEE 25th International Requirements Engineering Conference*, 1.

Liddy, E. D. (2001). Natural Language Processing . *Syracuse University*, 3.

Prakash M Nadkarni, L. O.-M. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 1.

IBM Cloud Education. (2020, July 2). *Natural Language Processing (NLP)*. Retrieved from IBM Cloud Learn Hub : https://www.ibm.com/cloud/learn/natural-language-processing

(n.d.). Retrieved from https://www.qualtrics.com/uk/experience-management/research/sentiment-analysis/

(n.d.).