

identifying A best model to analyse the sentiment experienced during Solitude

[Document subtitle]



[Date]

[Company name]

[Company address]

# Table of Contents

[Abstract 3](#_Toc115448443)

[Introduction 4](#_Toc115448446)

[LITERATURE REVIEW 5](#_Toc115448450)

[2.1.Multilingual sentimentanalysis: from formal to informal and scarce resource languages 5](#_Toc115448451)

[2.2.SOLO:A Corpus of Tweets for Examining the State of BeingAlone 6](#_Toc115448452)

3.[DATA SET 7](#_Toc115448455)

[3.1. Participants 7](#_Toc115448456)

[3.2. Collecting the Data 8](#_Toc115448457)

[3.3. Transcription of Voice Recording 9](#_Toc115448458)

[3.4. HARPA transcription conventions 11](#_Toc115448458)

[3.5. Meta data 11](#_Toc115448458)

[3.6. Text Mining 12](#_Toc115448458)

[3.7. Data wrangling 14](#_Toc115448458)

4. [METHODOLOGY 16](#_Toc115448459)

[4.1.VADER MODEL 16](#_Toc115448460)

[4.1.1 Sentiment Lexicons 16](#_Toc115448461)

[4.1.2.Machine learning approaches 17](#_Toc115448462)

[[4.1.3. Quantifying the emotion of a word 18](#_Toc115448463)](#_Toc115448462)

[4.1.4. Quantifying the emotion of a sentence 18](#_Toc115448462)

[4.1.5. Five simple heuristics 19](#_Toc115448462)

[4.2. BERT MODEL 21](#_Toc115448464)

[4.2.1. Architecture 22](#_Toc115448464)

[4.2.2. How BERT work? 24](#_Toc115448464)

[4.2.3. Input/Output Representation 25](#_Toc115448464)

[4.2.4. Masked Language Modeling 25](#_Toc115448464)

[4.2.5. Masked Language Modeling (MLM) 26](#_Toc115448464)

[4.2.6 Fine Tuning Procedure 26](#_Toc115448464)

[5. MODELING 27](#_Toc115448464)

[5.1. Vader model 27](#_Toc115448464)

[5.2. Bert model 29](#_Toc115448464)

[6.RESULT 29](#_Toc115448464)

[6.1.Vader result 29](#_Toc115448464)

[6.2. Bert result 30](#_Toc115448464)

[7. LIMITATIONS & FUTURE WORK 32](#_Toc115448464)

[8. CONCLUSION 32](#_Toc115448464)

**ABSTRACT**

**Objective:** People experience a wide range of emotions when they are alone. Forceful loneliness is regarded as being lonely and is frequently accompanied by sadness and anxiety. People typically seek solitude for self-reflection or as a break from the stresses of life. People experience a roller coaster of emotions during this time; some feel happy because of the tranquilly or peace around them while on vacation from their stressful lives, while others feel anxious, depressed, or even regret their life choices. It will be difficult for many models to analyse the sentiments because people feel a lot of different emotions when they are alone. The goal of this project is to find the best model that can analyse feelings of solitude accurately. To determine the best model for the analysis of the effects of solitude on people from various backgrounds, we will interpret each model and compare them.

**Methods:** The data set included 50 undergraduate students from Metro Vancouver who were between the ages of 18 and 28 (M = 20.0, SD = 1.8) and 100 community residents who ranged in age from 50 to 85 (M = 67.0, SD = 8.7). Each participant was given ten days to complete approximately thirty daily life assessments that focused on the individual's existing and ideal social situation, sentiments, and impact, as well as reactions.

**Results:** Althoughvader’s Lexcion approach is best for analysis the sentiment, vader cannot handle large data as the polarity score tends to move towards +1 or -1. At such time, its difficult to analyse and predict the sentiment. Vader only work when the data set is light and short.

**Conclusion:** Even though BERT takes lots of computational power and timing its appropriate model for heavy and long data. With small hyper tuning in model, shows greater degree of change in outcome for small data set. Overall BERT is the best model when it comes to analyse the sentiment while experiencing many emotions at once.

**1. INTRODUCTION**

The feeling of being alone is a common occurrence and everyone's experiences with alone time vary greatly. Humans experience solitude for a variety of reasons throughout their lifetimes, and they subjectively react to solitude in a wide range of ways that have a variety of effects. Some people may seek solace in solitude to escape the stresses of life, to engage in quiet reflection, to stimulate their creative urges, or to connect with nature. Others might be forced out of social situations or experience the pain and loneliness of social isolation, withdrawing. Psychologists distinguish between the positive state of voluntary aloneness known as solitude and the negative state of dissatisfaction with the calibre of one's social interactions known as loneliness. It has been demonstrated that loneliness—a negative and unwanted state of being alone—is associated with faster cognitive deterioration, dementia, depression, suicidal thoughts, self-harm, and even death. Contrarily, studies have shown that solitude—a constructive and self-driven state of solitude—improves autonomy, creativity, and wellbeing (Long et al., 2003; Knafo, 2012; Coplan and Bowker, 2017; Coplan et al., 2019a).

Now, we know that people go through many emotions at once, to analyse this we adaptive model to predict the sentiment. The process of analysing sentiment is called sentiment analysis. Within the realm of natural language processing, one of the most active research subfields is known as "opinion mining," which is another name for sentiment analysis. It considers the viewpoints of various individuals, feelings, evaluations, attitudes, and emotions by analysing how computers handle subjectivity in text (Hutto and Gilbert).

Techniques for Sentiment Classification can be loosely categorized as machine learning-based, lexicon-based, or hybrid. The Machine Learning (ML) Approach employs renowned ML techniques and language characteristics. The Lexicon-based Approach is dependent on a sentiment lexicon, which is a compilation of defined and precompiled sentiment phrases. It is subdivided into dictionary-based and corpus-based methods, which use statistical or semantic techniques to determine sentiment polarity. We will use Lexicon based approach to analyse to the sentiment of given data set.

Text classification methods that use ML can be roughly put into two groups: supervised learning and unsupervised learning. The method that we choose to proceed is based on the type of the data set. If the data set does not have any labels then we have to go with Unsupervised method hence the data set is unsupervised data set. If the data set contains labels for the records, then we will choose supervised method hence the data set is supervised.

Our data set is unsupervised. However, in order to use certain advance model we have to convert the unsupervised data set into supervised data set. The process of convert the data set is explain briefly below.

We are using HARPA (Health And intergenerational Activities Research Project) data set over other data sets is due to the collection data allowed the participant to express their feeling in higher degree.

Many natural language processing tasks have been proven to benefit from language model pre-training. These include sentence-level tasks like natural language inference and paraphrasing that aim to predict the relationships between sentences by analysing them holistically, as well as token-level tasks like named entity recognition and question answering that require models to produce fine-grained output at the token level.

Because of this big advantage of the token based model we choose to BERT pre-Trained model as another model to compare with, in this work.

**2. LITERATURE REVIEW**

**2.1. Multilingual sentiment analysis: from formal to informal and scarce resource languages**

The author (Lo et al.) discusses various methods and programmes currently in use for multilingual sentiment analysis, points out difficulties in this area of study, and offers several suggestions, including a framework that is especially useful for dealing with languages with limited resources. Subjectivity and polarity detection are the two main methods used in sentiment analysis. Understanding whether the content contains subjective viewpoints and opinions as opposed to information is the goal of subjectivity detection. On the other hand, polarity detection involves examining subjectivity using various polarities, intensities, or rankings. According to certain polarity analysis studies, a viewpoint could be highly positive, positive, negative, or highly negative. The majority of subjectivity and polarity analysis studies have focused on English-language content, but as online social media usage grows across the globe, this approach is no longer sufficient. In actuality, only 28.6% of Internet users are English speakers. Therefore, it is crucial to research or develop tools and resources in languages other than English. In addition, Asia now has 48.2% of all Internet users, followed by Europe (18%). As a result, improving languages like Chinese and Japanese is becoming more and more important.

Research on multilingual subjectivity and polarity analysis has become more common, and many other languages, such as Chinese, Japanese, German, Spanish, French, and Italian, have also been studied. The author discusses a variety of approaches for multilingual sentiment analysis, each of which is tested on a different set of data. These include translation-based polarity analysis, concept-based polarity analysis, lexicon and machine learning-based polarity analysis, corpus and machine learning-based polarity analysis, cross-lingual and machine translation polarity analysis, and many others. Overall, the author comes to the conclusion that the polarity-based lexicon-based approach is more effective when the dictionary or corpus of the weighted words of the particular language is current and includes regional slang.

However, our data has been carefully translated to English from mandarin and Chinese. Hence, we have to use normal Vader lexicon approach model works with the help of formal English dictionary.

**2.2 SOLO: A Corpus of Tweets for Examining the State of Being Alone**

The author (Kiritchenko et al. 2020) used the Twitter API to gather data on loneliness, loneliness, and solitude. The collected data set of tweets about solitude and loneliness excludes duplicate tweets and tweets with fewer than three words. The data set is known as the SOLO Corpus, which is an abbreviation for the State of being Alone corpus. The same author compiles a second set of general tweets from May to June 2019 by utilising the Twitter API and English function words (e.g., is, on, they, etc.). Two methods were applied to these data to analyse the emotions connected to the SOLO. The PMI method, which stands for pointwise mutual information, is based on the understanding that the best way to judge the relationship between two words is to ask whether or not they occur together more frequently than we would have a priori assumed they would by chance in a corpus. (Kiritchenko et al., 2014; Clark et al., 2016). The author says the corresponding association score for a word and a sub-corpus is greater than or equal to 1.5, then it is a strong association. The NRC Word-Emotion Association Lexicon is another approach and contains entries for more than 14,000 common English words. Eight fundamental emotions (angry, fear, sad, disgust, joy, anticipation, surprise, and trust) and two sentiments are given names (positive and negative). The binary labels state whether a word is connected to an emotion (or sentiment). Crowdsourcing the annotations allowed for the creation of the lexicon. The project  only take into account SOLO words that are lexically present, and it is calculate the proportion of words for each emotion (i.e., how many words out of every 100 are associated with sadness, joy, etc.) (Kiritchenko et al. 2020).

The limitation of the above work is, it is almost similar to Unidirectional. In our project will use both Lexicon based approach and Bi-Directional Approach and compare both models and interpret the best model using the accuracy and F1 score.

**3. DATA SET**

**3.1 PARTICIPANTS:**

The data set was collect on 100 community-dwelling people aged 50 to 85 (M = 67.0, SD = 8.7) and 50 undergraduate students aged 18 to 28 (M = 20.0, SD = 1.8) from Metro Vancouver were recruited. To increase statistical power and reflect people of all ages and backgrounds, we blended the two samples. Students were recruited through a university research subject pool, and older persons were attracted using community organisations, posters, referrals, and a database. Additionally, information from the sample of older adults was used in other studies (Lay et al., 2018; Pauly et al., 2018).

The older adult sample have a gender ratio of 64% women, a 56% East Asian, a 36% European, and 8% other/mixed origin. 72% of the sample had completed some form of college education. 92% of the students in the sample were female, 42% were East Asian, 22% were European, and 20% were of other or mixed ethnicity.  57% of older individuals and 28% of students were In a romantic relationship, and both samples had good health (M = 3.2 on 5-point subjective health ratings). There is a difference in remuneration between the two samples it is due to the fact that older persons additionally attended a 6-month follow-up session, whilst students did not. (Lay et al.)

The majority of older adult participants (57%), Mandarin (28%), and Cantonese (15%) completed the study. Student participants performed the study in English.

**3.2. COLLECTING THE DATA:**

A baseline session, a time-sampling period, and an exit session made up this study. The  Participants are told to filled out questionnaires measuring individual differences (such as trait self-reflection) and got instruction in the use of portable electronic devices at the baseline session. Participants were then beeped three times each day for ten days, starting the day after the baseline session (once in the morning, once in the afternoon, and once in the evening). Using a touch screen interface on an iPod or iPad mini, participants on each occasion answered a brief questionnaire about their thoughts, feelings, and current and ideal social situations. Beeps were timed with at least 4 hours between them in order to avoid conflicts with already set obligations.

Participants were asked, "What were you just thinking about?" and told to recorded their brief response on the keyboard or voice recorder in response to each beep. Then, they answered eight questions about their current thinking using a 100-point scale (0 being absolutely false, 100 being completely true). These questions were adapted from tests of mindfulness and reflection and rumination (Trapnell & Campbell, 1999). (Baer et al., 2006). The participants' current affective and cognitive-emotional states were measured with the next 12 items using a 100-point scale (0 = not at all, 100 = very lot).

Items were divided into four affect packages, each of which represented an affect with a high arousal (two items: "I am joyful," "I am thrilled," M = 36.8, SD = 28.3; "I am tranquil," M = 61.1, SD = 25.2; and (b) low arousal positive affect, consisting of three items: (c) high arousal negative affect (four items: "I am nervous," "I am satisfied," M = 68.7, SD = 24.4, "I feel close to others," M = 53.6, SD = 29.1) "I'm annoyed," M = 30.6, SD = 29. I feel shy, M = 23.9, SD = 27.3. (d) Low arousal negative affect (three items: "I am sad," M = 22.7, SD = 25.9; "I am tired," M = 46.0, SD = 32.3; and "I am lonely," M = 23.3, SD = 26.8). M = 16.0, SD = 20.9; "I am worried about what other people might think of me," M = 25.6, SD = 29.3; and "I am worried about what other people might think of me."

Now, collected data is in unstructured format, we have to convert it to structure format or at least in semi-structured format. The conversion process starts with the Tran scripting the Chinese or Mandarin of Video/Voice Recordings.

From below Figure 1, we see that nearly one third of participants have recorded their expression instead of keyboard. Hence, it is very important to Transcript the recording.

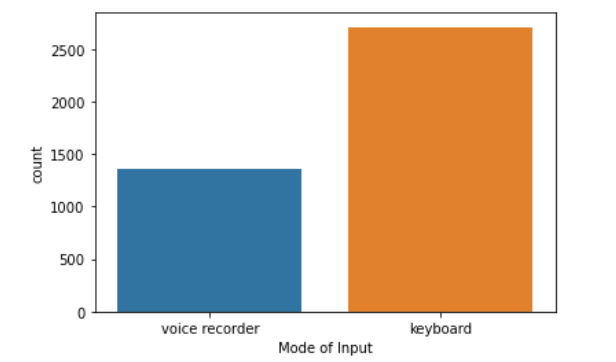


Figure 1: Comparisons of Mode of input

**3.3 Transcription of Voice Recordings:**

The transcription process for the vice recording involves five steps. As these steps going to produce the  text data for our model, these steps are crucial. A few conventions are followed when Tran scripts the recordings which is HARPA Transcription conventions; these conventions have meaning and emphasise the participant's speech when reading the Transcript. Only voice recordings qualify for these steps; keyboard entries are skipped in the case of other types of data.

Step 1: Person 1's initial transcription :Person 1 by Utilizing the transcription conventions (\*, [], etc.) Transcript the recording by listening the voice recording and enter the Transcript in word document. The participant ID number, voice recording file name, and the phrase "unverified transcription" are also entered on the word document and the document is saved in the "1unverified transcriptions" subfolder.

Step 2: Verification of transcription (Person 2): Unverified transcription from the step1 was copied into a new Word document and place it in the "2verified transcriptions" subfolder. The participant number, voice recording file name, and "verified transcription" also entered on the word document. If there, is any necessary changes to the transcription text and notes. The Changes will be done in "verified transcription" word file while listening to the voice recording.

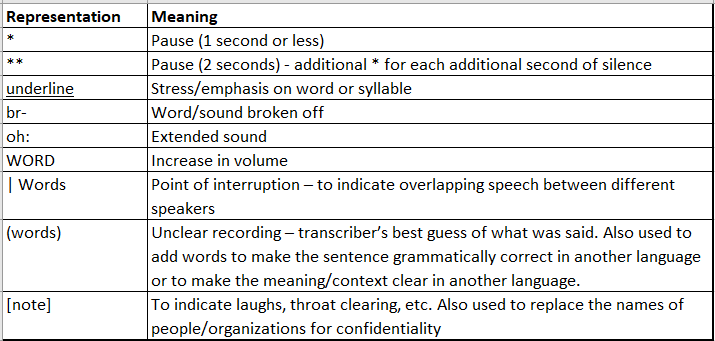
For next three steps same Person A who spent the majority of their time transcribing this participant's voice for voice-recorded data can be used. Person B should ideally be a different person from the participant's transcription team.

Step 3: Forward translation, Chinese to English (Person A): Verified transcription from Step 2 was copied into new document and place it in the "3. Forward translations" subfolder. Person A completes the forward translation from Chinese to English by listening to the recordings and the name of the voice recording, the participant's number, and the phrase “forward translation" are all included in the document's title. When finished, Chinese text is been deleted

Step 4: Backward translation, English to Chinese (Person B): The forward translation from Step 3 is copied into a new document, which is then saved in the "4. Backward translations" subdirectory. The name of the voice recording, the participant's number, and the phrase "backward translation" are all included in the document's title. Using only the English text as a guide, the text was translated from English to Chinese. Neither the voice recording nor any files containing the Chinese text are used.

Step 5: Final comparison to produce correct English version (Person A or Person B): The forward translation from Step 3 is copied into a new document, which is then saved in a subfolder called "5. Final comparisons” and the name of the voice recording, the participant's number, and the phrase “final comparisons “are all included in the document's title. Open the Chinese document that was reverse-translated in Step 4 and the authenticated Chinese transcription from Step 2. Verify that the text in these two documents is identical. Feedback was taken from all Translator to iron out any discrepancies to create a final English translation, modifications if it is necessary are done to the "final English" document.

**3.4. HARPA TRANSCRIPTION CONVENTIONS:**



**3.5. META DATA:**

In our data set, there are 88 variables (columns). We will go over a few of the variables it contains,

* ***ParticipantID*** - Participant ID
* ***Language*** - In which language did the participant provide their thought recordings? {1.00, English; 2.00, Mandarin; 3.00, Cantonese}
* ***QuestionnaireDate*** - Date when participant started the questionnaire using idialogpad
* ***ThoughtSamplMODE\_1\_CodeTR01, ThoughtSamplMODE\_2\_CodeTR01, ThoughtSamplMODE\_3\_CodeTR01-*** Thought-sampling input mode { Voice Recording; Keyboard Entry }
* ***ThoughtSamplTEXT\_1\_CodeTR02, ThoughtSamplTEXT\_2\_CodeTR02, ThoughtSamplTEXT\_3\_CodeTR02 –*** Thought-sampling input text (Entries and Voice Recordings has been Translated and Tran scripted and then add into final data set).
* ***ThoughtSampl\_1\_RecordingDate, ThoughtSampl\_1\_RecordingTime, ThoughtSampl\_2\_RecordingDate, ThoughtSampl\_2\_RecordingTime, ThoughtSampl\_3\_RecordingDate, ThoughtSampl\_3\_RecordingTime -*** Thought-sampling input voice recording date and Time.
* ***T01, T02, T03, T04, T05, T06, T07 -*** I was thinking about something that happened in the past (0-100); I was thinking about something happening in the future (0-100); My thoughts were mainly about myself (0-100) ; My thoughts were pleasant (0-100); I was having a hard time shutting off negative thoughts (0-100); I was just watching my thoughts go by without getting caught up in them (0-100); I was exploring new or 'deep' ideas (0-100); How difficult was it to describe your thoughts just now? (0-100), respectively.
* ***T08, T09, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21*** – Measuring the participants feeling and emotion through point scale of 0 -100 such as Happy, Calm, Sad, Anxious, Irritated, Satisfied, Feeling shy, Excited, Tired, Pain, Feeling Lonely, Worrying about Judged by others, Feeling close to someone, respectively.
* ***T01\_rt, T02\_rt, T03\_rt, T04\_rt, T05\_rt, T06\_rt, T07\_rt, T08\_rt,T09\_rt, T10\_rt, T11\_rt, T12\_rt,T13\_rt, T14\_rt, T15\_rt, T16\_rt,T17\_rt, T18\_rt, T19\_rt, T20\_rt,T21\_rt*** – it shows many milliseconds are taken by participants to answer point scale.

**3.6. TEXT MINING**

The process of converting unstructured text into a structured format for the purpose of identifying significant patterns and fresh insights is known as text mining, also known as text data mining. There are three type of data’s: Structured data, Unstructured data and Semi-structured data.

*Structured data* is tabular, with rows and columns, making it easier to store and process for analysis and machine learning. Names, addresses, and phone numbers are structured data inputs.

*Unstructured data* has no set format. It can include social media, product reviews, or video and audio files.

*Semi-structured data* combines structured and unstructured formats. It's organised, but not enough for a relational database. XML, JSON, and HTML files are semi-structured.

Our data is Semi-structured as it has organised and has meta-data for the dataset. Text pre-processing is done to get the text data ready for model building. It represents the initial phase. Pre-processing steps include, among others: Deleting punctuation such as (.,! $( ) \* % @), Deleting URLs, Removing Stop phrases, Lowering casing, Tokenization, removing HTMLS tags if there is so, Stemming and Lemmatization (Deepanshi).

Special characters in our data are mostly during Transcription of video recording and audio recording. As all special characters have some meaning and shows the participants true intention and hidden context of recording of the voice. However, we cannot use these special characters to build the model, as algorithm does not know the meaning of convention of the special characters. VADER model have different heuristics to analysis the sentiment of the text. And in BERT model all the heuristics produce less s

First, we have to convert the all words and letters in our input data to lower case. We are converting this from upper case as because we are using BERT Uncased base model for analysis the text. Hence, the model only accept the lower case text as input.

The next step is to remove all the punctuation from the text. We are doing this by using the function called punctuation from the library called string. Following this, we will remove all the white spaces, remove special character, the name, numbers from the text data. We use function from the text\_preprocessing library. By doing, this step we will remove all the special character that is added in the input text data in the Transcription process. The next step involves in removing stop words which are useless (such as “the”, “a”, “an”, “in”). We do not want these words to take up any unnecessary storage space or processing time in our data set. By keeping a list of the words you believe to be stop words, we can easily remove them for this reason. We will import nltk library and from it we import stopword function (“Removing Stop Words with NLTK in Python - GeeksforGeeks”). Following the above step, we will lemmatization the words. Lemmatization is the process of reducing a given word to its root word. Examples of lemmatization - ( rocks : rock; corpora : corpus ; better : good). For this we use the function call WordNetLemmatizer (“Python | Lemmatization with NLTK - GeeksforGeeks”).

We pass the input text data through all the functions mentioned above .We call all the above function through text\_preprocessing\_pipeline function that we created. The Final clean data is stored in the data set under the column name preprocessed\_text. We use the pre-processed data. Then, we have to correct the misspelled words or typing mistakes in our text data, so that word make sense. We will use function called Speller, which is imported form the autocorrect library to correct the text data. Through the removal of words that don't contribute to any model training operations, we are able to increase the efficiency of our processed content.

**3.7. DATA WRANGLING**

The initial step of the data wrangling is feature selection. We among 88 variables (columns) we have remove many variables, as it will not require for our analysis. We removed column like “QuestionnaireDate”; “ QuestionnaireTime”; “ThoughtSamplMODE\_1\_CodeTR01”; “ThoughtSampl\_1\_RecordingDate”; “ThoughtSampl\_1\_RecordingTime”; “ThoughtSampl\_2\_RecordingDate”; “ThoughtSamplMODE\_2\_CodeTR01”; “ThoughtSampl\_2\_RecordingTime”; 'ThoughtSamplMODE\_3\_CodeTR01', 'ThoughtSampl\_3\_RecordingDate'. Which tells about the Date and time of recordings or Keyboard entry in the Ipad. We only are only using Participant ID, Language they speak, Input text in all three columns, T09 (HAPPY), T10 (CALM), T15 (SAD) apart for these coloumn all other columns are drop. After that, we remove the entire row that have no recordings or no entry. By manual checking we noticed that some observation in the input text column haves “[no entry]”, "[translation pending]", “(no description provided)", "nothing". We have also removed the entire row from the data set. Next Step, we are combining all three-text columns into one column. Following this, we have renamed the column T09, T10, T15 into happy, calm and sad respectively. As we have few null values in above three columns, we remove it from the data set.

Because the data set that we are using is an unsupervised data set, we are going to change the data set into a supervised data set by labelling records using a 100-point scale measurement of happy, sad, and calm, with 0 indicating not at all and 100 indicating a great deal. We are doing this because BERT is supervised model. The measurement of the point scale response given by the participant can be used to label the text, which is a logical way to proceed. If the measurement in the HAPPY column is higher than the measurements in the SAD and CALM columns, then the record is labelled as HAPPY. The measurements in the same other two columns are compared and labelled according to their respective values. This approach to labelling the text does have one limitation, and that is when participant response point scale measurement for Happy is 25, for SAD it is 20, and for CALM it is 18. In this case, the approach that we discussed earlier would cause the text to be marked as happy, even though the participant might not actually be feeling happy or the individual might be feeling awful. This limitation can be circumvented by incorporating an additional layer of logic into the method, which entails maintaining a threshold for the value of each column. If the value of the column is greater than the threshold, then the column is labelled; otherwise, it is eliminated. Now we are going to convert the words happy, sad, and calm into their corresponding numerical values, such as 2 for happy, 1 for sad, and 0 for calm. Because string labels will cause the BERT model to become confused, we are converting them into numerical form.

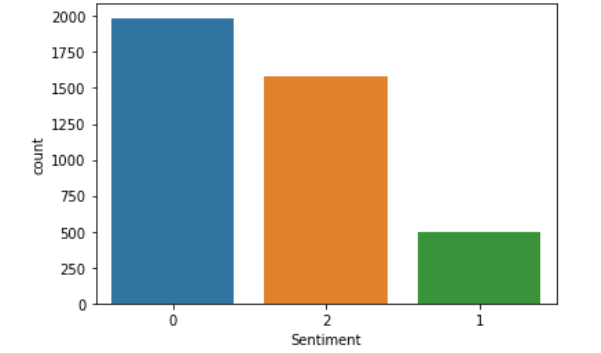


Figure 2: Distribution of all three sentiments

After convert the labels a graph has been plotted as shown in figure 2. The Graph show the distribution of the sentiment in the data set. From the plot it seems like, people who feel sad is very low compare to other two sentiments.

**4. METHODOLOGY**

**4.1 VADER MODEL**

VADER, a simple rule-based model for general sentiment analysis, For fast, accurate sentiment analysis on such a large scale, you often need a large and high-quality dictionary. The Linguistic Inquiry and Word Count (LIWC, pronounced "Luke") is an example of this kind of dictionary that has been used a lot in social media (Pennebaker, Francis, & Booth, 2001; Pennebaker, Chung, Ireland, GonMzales, & Booth, 2007).

VADER, which stands for *Valence Aware Dictionary for Sentiment Reasoning,* and  we will study briefly about how it was made, tested, and judged.  A mix of qualitative and quantitative methods to create a gold-standard sentiment lexicon that is especially well-suited to microblog-like contexts, which we then test in the real world.

Next, we combine these lexical features with five rules that can be used to describe grammatical and syntactic conventions that humans use when expressing or emphasising the intensity of their feelings. It is found that putting these heuristics into the sentiment analysis engine makes it more accurate in a number of domains, including social media text, movie reviews, and product reviews. The VADER lexicon does very well in the social media domain, which is very interesting. When it comes to matching ground truth, the correlation coefficient reveals that VADER (r = 0.881) performs just as well as individual human raters (r = 0.888). When we look more closely at the classification accuracy, we find that VADER (F1 = 0.96) does a better job than human raters (F1 = 0.84) at correctly putting tweets into positive (Hutto and Gilbert). Vader have two approaches Sentiment lexicons (Polarity-based and Valence-based) and Machine Learning approach (Naive Bayes and Support Vector Machines).

**4.1.1. SENTIMENT LEXICONS**

A large number of approaches to analysing people's feelings depend heavily on a lexicon of feelings or opinions. A sentiment lexicon is a list of lexical items (like words) that are usually labelled as either positive or negative based on how they are used (Liu, 2010).

Manually creating and validating such lists of opinion-bearing features is one of the best ways to make reliable sentiment lexicons, but it is also one of the slowest. Because of this, a lot of the applied research that uses sentiment analysis relies heavily on lexicons that have already been built by hand. Because lexicons are so helpful for figuring out how people feel, we'll quickly go over a few benchmarks. First, we look at three popular lexicons (LIWC1, GI2, and Hu-Liu043) in which words are put into either positive or negative classes based on their meanings outside of context (Hutto and Gilbert).

**4.1.2. MACHINE LEARNING APPROACHES**

Since manually compiling and validating an extensive sentiment lexicon takes a lot of time and effort, many studies have looked into automated techniques for finding sentiment-relevant textual elements. Modern best practises use machine learning techniques to "learn" the text's sentiment-relevant elements. (Hutto and Gilbert). The Naive Bayes (NB) classifier is a straightforward algorithm that uses Bayesian probability and the naive assumption that feature probabilities are independent of one another. This makes the NB classifier an unsupervised learning algorithm. Maximum entropy, also known as MaxEnt or ME, is a method of machine learning that is used for a variety of applications and is classified as an exponential model that employs multinomial logistic regression. ME, in contrast to NB, considers information entropy and does not assume conditional independence between features (feature weightings). (Hutto and Gilbert).

Support Vector Machines (SVMs) are non-probability classifiers that distinguish themselves from both NB and ME models by dividing data points in space using one or more hyperplanes (centerlines of the gaps separating different classes). (Hutto and Gilbert). The Python-based machine learning techniques from scikit-learn.org are used for the NB, ME, SVM-Classification (SVM-C), and SVM-Regression (SVM-R) models. (Hutto and Gilbert).

There are disadvantages to machine learning techniques. They first require (sometimes large amounts of) training data, which can occasionally be difficult to obtain (as with validated sentiment lexicons). They also rely on the training set to include as many features as feasible, which is sometimes not the case, particularly with the brief and sparse language found on social media. Third, they typically have a higher overall cost in terms of the processing power of the central processing unit (CPU), the amount of memory that is required, and the amount of time spent on training and classification. Fourth, they frequently originate from aspects "behind the scenes" inside of a "black box" that are not  easy to interpret by humans and are, as a result, more difficult to generalise, change, or extend (for example, to other domains). (Hutto and Gilbert).

**4.1.3. QUANTIFYING THE EMOTION OF A WORD**

VADER sentiment analysis primarily uses a lexicon that converts lexical characteristics into sentiment scores, a measure of the strength of an emotion. By adding the intensity of each word in a text, one can determine the sentiment score of that text. However, what is Lexical feature and how do we gauge its emotional intensity? Lexical features of textual communication for example, Consider a tweet as an illustration. In addition to text, a typical tweet might contain emoticons like ":-)," acronyms like "LOL," and slang like "meh." These slang terms are mapped to intensity values as well, which is a neat feature of VADER sentiment analysis (Calderon).

On a scale of -4 to +4, with -4 being the most negative and +4 representing the most positive emotions, the sentiment score or emotion intensity is determined (Calderon). The score zero stands for a neutral opinion. "Horrible" and "okay," two examples of dictionary terms, are mapped to -2.5 and 0.9, respectively. The emoticons "/-:" and "0:-3" are also assigned to -1.3 and 1.5, respectively. The dictionary is build utilising rater’s from Amazon Mechanical Turk (Calderon).

You might believe that the degree of emotional intensity is highly subjective because it is dependent on the person being asked. While you might not consider certain words to have a particularly negative connotation, I might. In order to combat this, the people who developed the VADER sentiment analysis enlisted the help of more than one human rater and then took the average of all of their ratings for each word. This is based on the idea of the "wisdom of the crowd," which states that the opinions of a group are frequently more reliable than the opinions of an individual. Consider the television show "Who Wants to Be a Millionaire?" for example. Ask the Audience is one of the lifelines that contestants can use, and like the other lifelines, it relies on the collective intelligence of the audience (Calderon).

**4.1.4 QUANTIFYING THE EMOTION OF A SENTENCE**

The sentiment score of individual words ranges between -4 and 4, but the sentiment score returned for a sentence falls somewhere between -1 and 1. They are equally correct. A sentence's total sentiment score is equal to the sum of the sentiment scores attributed to its individual sentiment-carrying words. The sentiment score of a sentence is determined by adding together the individual sentiment scores assigned to each word in the sentence that is listed in the VADER dictionary. On the other hand, we perform a normalisation on the sum, which maps it to a value that ranges from -1 to 1.

C:\Users\NSK\AppData\Local\Microsoft\Windows\INetCache\Content.Word\0_jwZcRNigBLk1w1H0_.pngThe normalization used by Hutto is

Where x is the total number of words in the sentence and alpha is a normalisation parameter; x represents the sentiment scores of the individual words in the sentence. The graph that follows displays the

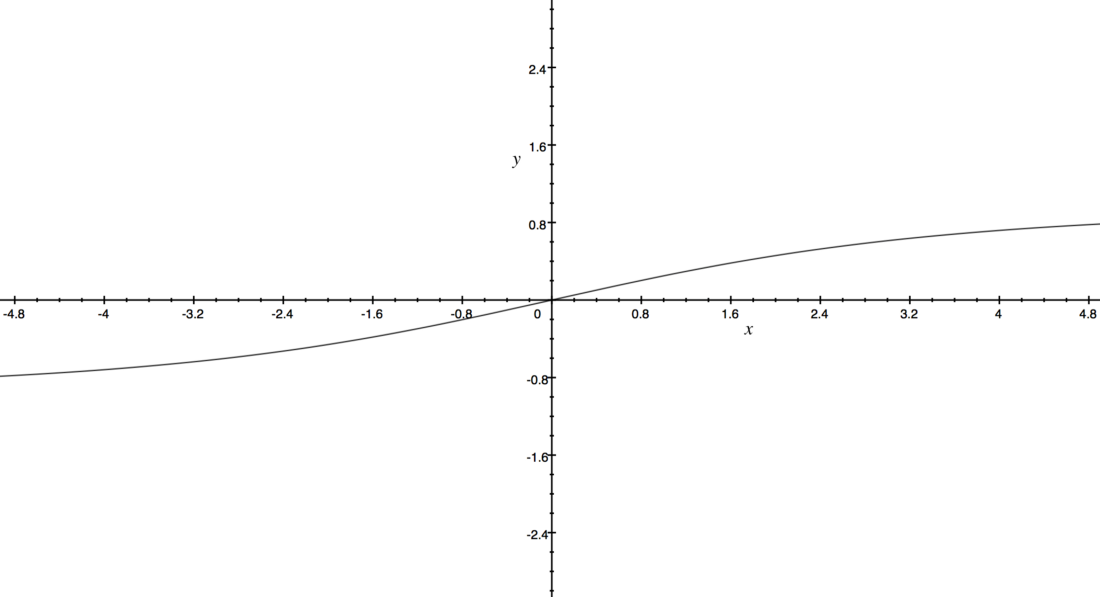


Figure 3: Normalize graph of VADER

From figure 3 we can see that as x increases, we can see that it is getting closer and closer to either -1 or 1, as shown here. To a similar effect, if the document that you are applying VADER sentiment analysis to contains a large number of words, you will receive a score that is close to -1 or 1. As a result, VADER sentiment analysis functions most effectively on short documents, such as tweets and sentences, rather than on lengthy documents. (Calderon). Our data set only have 4000 plus records hence this methodology is appropriate to use.

**4.1.5. FIVE SIMPLE HEURISTICS**

The sentiment conveyed by a sentence can be impacted by more than just the lexical components of the phrase. There are additional components of context, such as punctuation, capitalization, and modifiers, that also contribute to the expression of emotion. The VADER sentiment analysis takes all of these factors into consideration by using five straight forward heuristics. Once again, the effectiveness of these heuristics is measured with the help of human raters.

The use of punctuation is the first heuristic to consider. Compare "I like it." and "I like it!!!" It is not overly difficult to argue that the second sentence contains more intense emotion than the first, and that it must therefore have a higher VADER sentiment score because of this fact.

The VADER sentiment analysis takes this into consideration by increasing the sentence's overall sentiment score in proportion to the number of exclamation points and question marks that are used to end the sentence. First, VADER determines the sentiment score associated with the sentence. If the result is positive, VADER adds a specific amount (0.292) for each exclamation point and question mark (0.18). In cases where the score is already negative, VADER performs a subtraction (Calderon).

The use of capitalization is the subject of the second heuristic. The phrase "AMAZING performance" connotes a significantly higher level of intensity than the phrase "amazing performance." As a result, VADER takes this into account and either adds 0.733 to the sentiment score of the word or subtracts 0.733 from it, depending on whether the word is positive or negative (Calderon).

The utilisation of degree modifiers is the third piece of advice that can be taken. Consider the phrases "effing cute" and "sort of cute" as an example. In the first sentence, the effect of the modifier is to intensify the cuteness of the noun, whereas in the second sentence, the effect of the modifier is to lessen the cuteness of the noun. A booster dictionary is kept up to date by VADER. This dictionary includes a variety of boosters and dampeners (Calderon).

The proximity of the degree modifier to the word that it is modifying also plays a role in the effect that it has. The base word experiences an intensifying effect that is proportionately less pronounced when further words are added. If the base word is positive, adding one positive modifier to it raises the sentiment score of the sentence by 0.293 points; if the base word is negative, adding one negative modifier lowers the score by 0.293 points. Adding or subtracting 95% of 0.293 with a second modifier from the base word, and adding or subtracting 90% with a third modifier, respectively (Hutto and Gilbert).

The change in polarity that occurs as a result of the word "but" is the subject of the fourth heuristic. The word "but" is used to connect two clauses that have opposing ideas most of the time. The latter sentiment, on the other hand, is the one that predominates. As an illustration, one might say, "I love you, but I don't want to be with you any longer." The first clause, "I love you," expresses a positive sentiment; however, the second clause, "I don't want to be with you anymore," expresses a negative sentiment, which is obviously more dominant in terms of sentiment (Calderon).A "but" checker has been implemented by VADER. Simply put, the valence of all words that carry an emotion before a "but" is reduced to half of what it would normally be, while the valence of words that follow a "but" is increased to 150% of what it would normally be (Calderon).

The fifth piece of heuristics is to look at the trigram before a feature of the lexicon that is emotionally loaded in order to spot polarity negation. A lexical item that contains three distinct aspects is referred to as a tri-gram in this context. A list of negator words is kept up to date by VADER. Multiplying the sentiment score of the sentiment-laden lexical feature by an empirically determined value of -7.74 allows for the capture of negation in the analysis (Calderon).

**4.2. BERT MODEL**

Researchers at Google AI Language released a new methodology for Natural Language Processing (NLP) called BERT (Bidirectional Encoder Representations from Transformers) towards the end of 2018; it’s astounding performance shook the Deep Learning community to its core. In the pre-BERT era, a language model would have viewed this text sequence from either left-to-right or a combo of left-to-right and right-to-left during training. This strategy works well for creating sentences; we may anticipate the next word, append it to the sequence, and then predict the next-to-next word until a complete phrase is generated.

Researchers contend that current strategies limit the power of pre-trained representations, particularly fine-tuning approaches. The most significant drawback is that typical language models are unidirectional, which limits the architectures that can be employed during pre-training. For example, in OpenAI GPT, the authors employ a left-to-right design in which each token can only attend to preceding tokens in the Transformer's self-attention layers (Vaswani et al., 2017).

In contrast to current language representation models (Peters et al., 2018a; Radford et al., 2018), The goal of BERT is to pre-train deep bidirectional representations from unlabeled text by conditioning on context from both the left and the right in each layer. As a result, the pre-trained BERT model may be fine-tuned by just even adding one layer on it, which provides more sophisticated models, such as query answering and language inference, without any significant task-specific architecture tweaks.

On eleven different natural language processing tasks, BERT achieves new state-of-the-art results, in GLUE it scored 80.5%  and in  MultiNLI accuracy scored 86.7%  (Jacob Devlin).

**4.2.1Architecture**

Depending on the size of the model architecture, there are four different pre-trained BERT variants:

BERT-Base, Uncased: 12-layers, 768-hidden, 12-attention-heads, 110M parameters.

BERT-Large, Uncased: 24-layers, 1024-hidden, 16-attention-heads, 340M parameters.

BERT-Base, Cased: 12-layers, 768-hidden, 12-attention-heads, 110M parameters.

BERT-Large, Cased: 24-layers, 1024-hidden, 16-attention-heads, 340M parameters. (google-research)

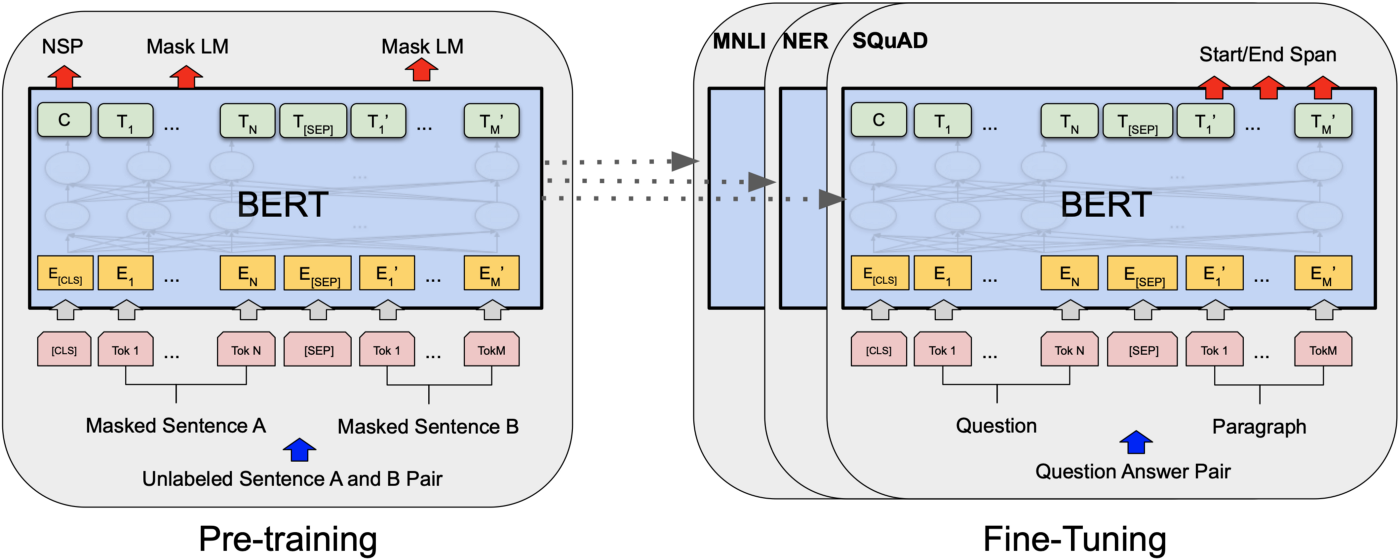


Figure 4: General procedure for pre-training and fine-tuning BERT. The same architectures are used for pre-training and fine-tuning, with the exception of output layers. Models are initialised for various down-stream tasks using the same pre-trained model parameters. All parameters are adjusted during fine-tuning.

BERT employs a cutting-edge method called Masked LM (MLM), which randomly masks words in the phrase before attempting to predict them. This method is used in place of traditional word prediction. When a word is "masked," the model uses both left and right surrounds, as well as both directions of the sentence, to anticipate the hidden word. It considers the previous and following tokens simultaneously, in contrast to earlier language models. This "same-time portion" was absent from the existing combined left-to-right and right-to-left LSTM based models. (However, it could be more correct to describe BERT as non-directional.)

Language representations that have already been trained can be either context-free or context-based. Consequently, context-based representations may be unidirectional or bidirectional. For each word in the lexicon, context-free methods like word2vec provide a single word embedding representation (a vector of numbers). The term "bank," for instance, might be represented as both "bank account" and "bank of the river" without regard to context.

The portrayal of each word in context-based models, on the other hand, is based on the meanings of the other words in the phrase. A unidirectional contextual model, for instance, would represent "bank" based on "I accessed the" but not "account" in the sentence "I accessed the bank account." BERT, on the other hand, deeply bidirectionally depicts "bank" utilising both its previous and next context — "I accessed the... account" — starting from the very start of a deep neural network.

The pre-training process substantially adheres to the body of literature that already exists on language model pre-training. We use the Books Corpus (800M words) (Zhu et al., 2015) and English Wikipedia for the pre-training corpus (2,500M words). For Wikipedia, we just extract text portions and leave off headings, tables, and lists. To extract lengthy sequences, it is necessary to use a document-level corpora rather than a shuffled sentence-level corpus like the Billion Word Benchmark (Chelba et al., 2013).

In addition, BERT uses the Transformer model architecture as its foundation rather than LSTMs. A Transformer operates by carrying out a limited number of repeated steps. It uses an attention mechanism in each step to comprehend the connections between all the words in a sentence, regardless of where they are in the sentence. The Transformer is able to learn to instantaneously pay attention to the word "river" and make this determination in just one step. For example, given the sentence, "I arrived at the bank after crossing the river," the Transformer can decide that the word "bank" refers to the shore of a river and not a financial institution, determined in one step (Jacob Devlin).

**4.2.2.How BERT work?**

BERT is dependent on a Transformer (the attention system that recognises how words in a text relate to one another in context). The essential components of a Transformer are an encoder, which reads the text input, and a decoder, which generates a prediction for the task. Since the objective of BERT is to produce a language representation model, just the encoder portion is required. The input to the BERT encoder is made up of a string of tokens, which are then converted into vectors before being passed through the neural network for processing. However, BERT requires the input to be modified and embellished with additional metadata before processing can begin:

Token Embedding: At the start of the first sentence, a [CLS] token is added to the input word tokens, and at the conclusion of each sentence, a [SEP] token is added.

Segment Embedding: Each token receives a marking designating Sentence A or Sentence B. Because of this, the encoder can tell which sentences are which.

Positional Embedding: Each token is given a positional embedding to show where it belongs in the sentence.

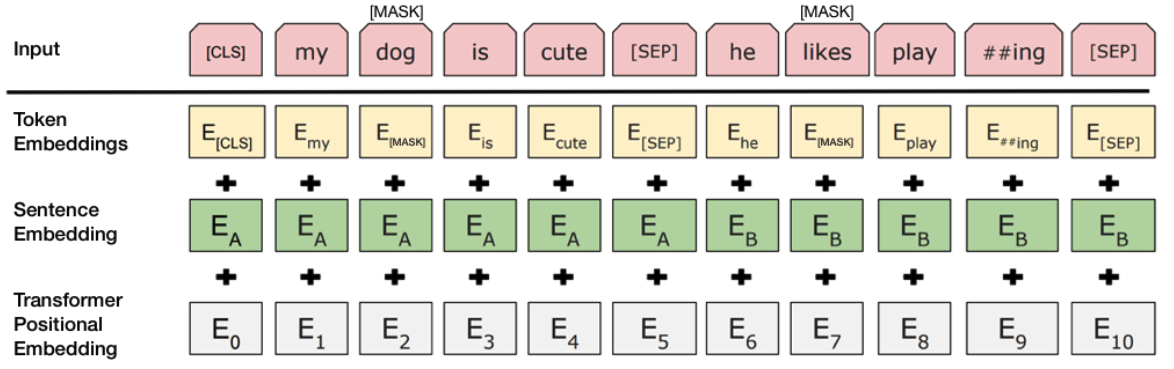


Figure 5: input representation for BERT. The input embedding is made up of the token, segmentation, and position embedding is added together. (Jacob Devlin).

The Transformer basically adds a layer that maps sequences to sequences, which results in an output which is also a series of vectors with input and output tokens correlating 1:1 at the same index. This results in the Transformer's ability to transform sequences into sequences.And as we already know, BERT does not attempt to forecast the word that will come after the current one. Two tactics are employed during training. (Jacob Devlin).

**4.2.3. INPUT/OUTPUT REPRESENTATIONS**

Our input representation can unambiguously represent both a single sentence and a pair of sentences in one token sequence, enabling BERT to handle a variety of down-stream tasks. With 30,000 tokens, researchers used Word Piece embedding’s (Wu et al., 2016). Every sequence always starts with a special classification token as the first token ([CLS]). For classification tasks, the token corresponding to the last hidden state is used as cumulative sequence representation. A single sequence contains several sentence pairs. We distinguish the sentences in two different ways. We first divide them using a unique token ([SEP]). In order to determine whether a token belongs in sentence A or sentence B, we secondly add a learned embedding to each token. The final hidden vector of the unique [CLS] token is denoted as C ∈ R^H in Figure 4, and the final hidden vector for the i-th input token is denoted as Ti ∈ R^H. The input representation for a given token is created by adding the corresponding token, segment, and position embedding’s. Figure 5 depicts this construction in visual form.

**4.2.4.Masked LM (MLM)**

Run the whole series through the BERT attention based encoder, mask 15% of the input words at random, and then predict only the masked words using the context supplied by the other non-masked words in the sequence. This naive masking strategy has a drawback in that the model only attempts to predict the correct tokens when the [MASK] token is present in the input, when what we really want is for the model to attempt to predict the proper tokens regardless of which token is present in the input. To solve this problem, 80% of the tokens out of the 15% chosen for masking are actually substituted only with token [MASK] (Jacob Devlin). 10% of the time, a random token is used to replace a token and Tokens are unaltered 10% of the time (Jacob Devlin).

The BERT loss function ignores the forecast of the non-masked tokens during training and only takes into account the prediction of the masked tokens. As a result, the model converges significantly more gradually than right-to-left or left-to-right models (Jacob Devlin).

**4.2.5.Next Sentence Prediction (NSP)**

The BERT training procedure additionally makes use of the next sentence prediction to comprehend the relationship between two sentences. When doing activities like answering questions, a pre-trained model with this kind of expertise is useful. The model learns to predict whether the second sentence is the next one in the original text when given pairs of sentences as input during training.

As we've already seen, BERT uses the unique [SEP] token to break up sentences. Two input sentences are provided to the model at once during training, with the second sentence coming after the first one 50% of the time. 50% of the time, a random sentence chosen from the entire corpus is used. (Jacob Devlin). In order to determine if the second sentence is random or not, BERT must first determine whether the random statement is connected to the first sentence (Jacob Devlin). Figure 6 shows an example for NSP flow by CLS and SEP.

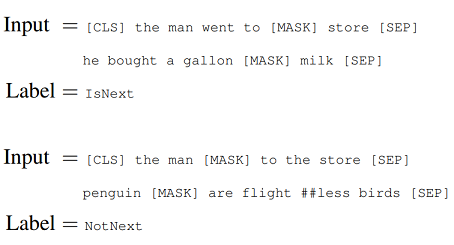


Figure 6: Example for NSP flow by CLS and SEP

The entire input sequence is processed by a Transformer-based model, the output of the [CLS] token is transformed into a 21-shaped vector by a simple classification layer, and the IsNext-Label is determined with the help of softmax. This is done so that we can determine whether or not the second sentence is connected to the first one (Jacob Devlin). Both Masked LM and Next Sentence Prediction are used to train the model. This is done so that the two strategies' combined loss function is as small as possible (Jacob Devlin)

**4.2.6.Fine-tuning Procedure**

Most of the model's hyper parameters are the same for fine-tuning as they were for pre-training, but the batch size, learning rate, and number of training epochs are different. The chance of dropping out was always kept at 0.1. (Jacob Devlin) The best values for hyper parameters depend on the task, but we found that the following range of values worked well for all tasks:

•Batch size: 16, 32

•Learning rate (Adam): 5e-5, 3e-5, 2e-5

•Number of epochs: 2, 3, 4

Authors also noticed that small datasets were much more sensitive to hyper parameter choice than large datasets (e.g., 100k+ labelled training examples). Fine-tuning is usually very quick, so it makes sense to just search through all of the above parameters and pick the model that works best on the development set. (Jacob Devlin)

**5. MODEL BUILDING:**

**5.1. BERT MODEL**

First, we download the pre-trained BERT mode. Since we don't have a Google TPU, we chose the BERT base uncased model with 12 layers, 768 hidden heads, 12 attention heads, and 110 million parameters. Then, we choose "cased" or "uncased" based on whether or not we think letter casing will help with the task (Khalid).

We split the data set into three parts naming Train set, validation set and test set. The Train set is split in ratio of 0.8. Remaining 0.2 percent is split into two equal part of 0.1 percent and assigned to the valid and test data set.

To use BERT, we must change our data into the format that BERT expects. Out of memory errors can happen when you train with BERT. Usually, this means we need more powerful hardware, like a GPU with more RAM or a TPU. We can try some workarounds, though, before we look into getting better hardware. For example, we can try to cut down on the amount of data used for training (training batch size), but this will slow down the training. (Khalid).

Next, it makes a single new layer that will be trained to adapt BERT to our sentiment task (i.e. classifying whether a Text is positive or negative). Fine-tuning is the process of using a model that has already been mostly trained.(Google Colab).

The BERT model's input data should be vectorised, hence we  encode the data before passing it though the model. Word Embedding’s, also known as Word Vectorization, is a technique used in natural language processing to map words or phrases from a vocabulary to a corresponding vector of real numbers. These vectors can then be used to find word predictions, word similarities, and word semantics. Vectorization is the process of converting words into numerical values (Prabhu). Both the train and valid data sets are encoded. We set attention mask hyper- parameter to true, this mask the less important words in the input text as shown in figure four. The padding max length is then set to true. Padding is a technique for ending each short input data with a zero. For example, when maximum length in the input data is 150, then when short input data is needed to pass, we add zero for remaining length of the short data. The vector of input data and vector of masked data and tensor format of label of the data is passed through the model.

The batch size that we use is three, and the epoch value is five. We set the batch size and Epoch to small values in order to decrease the amount of power it takes for the computation. An epoch is the once-through forward and backward propagation of the entire data set through all layers of neural network. It is possible to count both forward and backward propagation as a single iteration, but only when they occur together. For example,   we have 1000 data points; I will explain how batch size and the epoch parameter can reduce the amount of computational power needed. If the size of the batch is one thousand, then we will only need one iteration to finish an epoch. In a similar vein, an epoch is comprised of two iterations when the batch size is 500. If the batch size is 100, then the number of iterations required to finish an epoch is ten. Simply put, the number of data points can be calculated as the required number of iterations multiplied by the batch size. In this scenario, the neural network is provided with multiple instances of the same data.

After that, we train the model and store all of the necessary outputs and metrics for each epoch in the system. We use it to validate the accuracy of the model. We choose the best epoch with better metrics to predict the sentiment on test data.

**5.2. VADER MODEL**

It is a Lexicon-based model, as was already seen in the methodology of Vader. We invoke the SentimentIntensityAnalyzer function that is located in the nltk.sentiment.vader library. After that, text is passed through the model. In response, the model provides a polarity score for the data that was input. The polarity score of the first record that is used as input data is displayed in the figure 7. The Vader assigns a score based on all positive, negative, and neutral polarities. Based on our intuitive understanding, we ascribe a feeling of neutrality to calmness, given that calmness can mean either happiness or suffering (Courtney). The scores range from minus one to plus one. We assign labels such as "HAPPY," "SAD," and "CALM" to the polarity score based on the degree to which it is achieved. The category of "Sad" is assigned whenever the score is below zero. If the score is positive, then it is considered to be in the HAPPY category, whereas if the score is zero, then it is considered to be in the CALM category. This procedure is carried out once more for each complete record of input text, and the results are saved in the column designated by the name vader\_result.

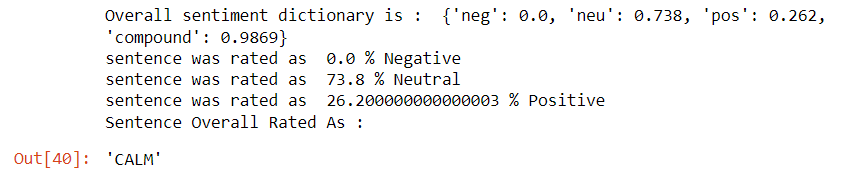
****

Figure 7: Output of Vader predicts for the first input data

**6. RESULT**

**6.1. VADER RESULT**

The accuracy of the Vader model is 48%. Due to the fact that our data set is imbalanced, we are unable to rely on the accuracy score to measure the performance of our model. When the classes are evenly distributed and there is little risk associated with predicting false negatives, accuracy is used. This is the case when there is no major downside to predicting false positives. The F1 score is a combination of precise driving and fast reaction times. The equation for determining the F1 Score is as follows: F1 Score = 2 \* (Precision \* Recall) / (Precision + Recall). We obtain the f1 score for each class (label) by using the below metrics shown in figure 8; the overall f1 score of the model is determined by taking the average of the f1 scores for all three labels. The overall model has a score of 0.551 for the f1 test.

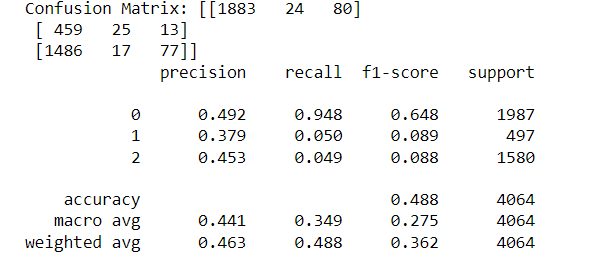


Figure 8 : Metrics of Vader Model

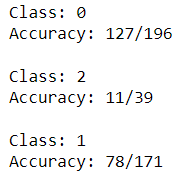
Reading the confusion matrix of the model is the only way for us to determine the areas in which our model performed particularly well. Due to the fact that our model forecasts three values, the confusion matrix has been formatted in a 3x3 grid. The matrix has nine cells and is read from left to right across its surface. The actual values are on the left-hand side of, and the predicted values are on the top. The first cell is a True positive, while the second cell is a false negative. The third cell is a false negative, the fourth cell is a false positive, the fifth cell is a true negative, the sixth cell is a false negative, the seventh cell is a false positive, the eighth cell is a false negative, and the ninth cell is a True negative. We are able to compute the accuracy, the f1 score, and all of the metrics using this matrix. The formula for determining an individual's f1 score is as follows: 2xTP /2xTP + FP + FN. The formula for precision is TP/(TP+FP), while the formula for accuracy is (TP+TN)/(TP+TN+FP+FN). The calculation for recall is TP divided by (TP+FN) (Mohajon).

All of the metrics are computed by first comparing the actual values (LABELS), and then using the polarity score to predict new values.

**6.2. BERT**

As mentioned above, we check the metrics of each epoch and choose one epoch with better scores to predict the sentiment in the test data set. Figure 9 show the metric of first five epochs We then calculate the accuracy of the prediction by comparing the actual value (LABELS) and predicted value. Accuracy for each class (labels) is mentioned below.

The accuracy is significantly higher than the Vader’s accuracy; we can still increase the accuracy by using lot more records for training the model.



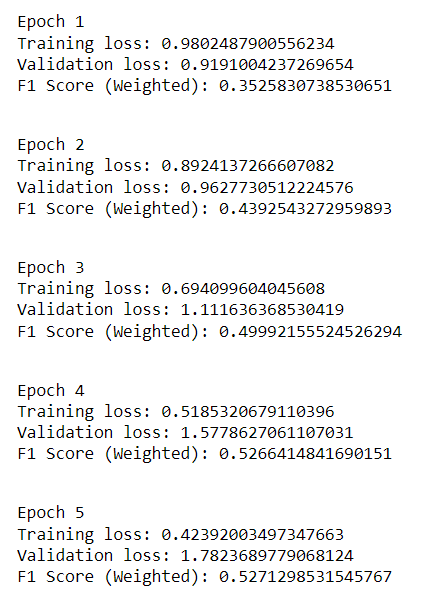
****

Figure 9: Metrics of BERT epoch training

**7. LIMITATIONS**

As a result, because our computational power is limited, we have no choice but to use the BERT base model, which has fewer neural layers. However, we only had a small data set to work with, and since BERT is a deep learning model, it needs a very large data set in order to show its full performance. We will able to circumvent this challenge by performing hyper tuning on the model, given that small twerks are capable of producing a higher degree of change for predicting the output having small data sets. Instead of only having labels for happy, sad, and calm, we could have increased the number of labels and used the BERT model to analyse them; the BERT model will be able to easily handle this situation. Due to the fact that we are contrasting the BERT model with the Lexicon-based Vader model, we are only allowed to use three labels. This is because Vader can only evaluate three types of emotions: positive, negative, and neutral. When comparing different models, it is only fair to use the same input for all of the models in each comparison.

We can use same methodology and use “HIDDEN MARKOV MODELS” for analysing the different solitude experience between South East Asians and Western peoples. Due to the time limits and computational power constrains, we were not able to build the model and analyse the difference between two culture peoples. However, using Google colab syetem for training model we can achieve the expected output.

**8. CONCLUSION:**

When compared to methods such as exploratory analysis and the Point-wise mutual information approach to analyse sentiment, the Vader model stands out among the other models mentioned above because it classifies sentiment into three distinct classes. However, Vader is unable to perform adequately when it comes to the management of large and lengthy data sets. Because there is not much variation in the polarity score, which tends to be either close to+1 or -1. Above all else, Vader is unable to analyse multiple feelings at once, such as happiness, anxiety, depression, loneliness, and so on. Vader is best suited for analysing formal sentences as opposed to informal ones or text data that contains slang. BERT, on the other hand, is able to analyse multiple feelings at once (using multi-label classification), and it can manage large and lengthy data sets. Additionally, BERT demonstrates better results than Vader models, albeit at the expense of a significant amount of computational time and power. Therefore, when it comes to analysing multiple feelings that are experienced during periods of solitude, the BERT model is the most effective method.

**REFERENECE**

1. Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge,Thorsten Brants, Phillipp Koehn, and Tony Robin-son. 2013. One billion word benchmark for measur-ing progress in statistical language modeling.arXivpreprint arXiv:1312.3005.
2. Matthew Peters, Mark Neumann, Mohit Iyyer, MattGardner, Christopher Clark, Kenton Lee, and LukeZettlemoyer. 2018a. Deep contextualized word rep-resentations. InNAACL.
3. Alec Radford, Karthik Narasimhan, Tim Salimans, andIlya Sutskever. 2018. Improving language under-standing with unsupervised learning. Technical re-port, OpenAI.
4. Pennebaker, J. W., Chung, C. K., Ireland, M., Gonzales, A., & Booth, R. J. (2007). The development and psychometric properties of LIWC2007. Austin, TX: LIWC.net
5. Pennebaker, J. W., Francis, M., & Booth, R. (2001). Linguistic Inquiry and Word Count: LIWC 2001. Mahwah, NJ: Erlbaum
6. Liu, B. (2010). Sentiment Analysis and Subjectivity. In N. Indurkhya & F. Damerau (Eds.), Handbook of Natural Language Processing (2nd ed.). Boca Raton, FL: Chapman & Hall.
7. Lay, J. C., Pauly, T., Graf, P., Mahmood, A., & Hoppmann, C. A. (2018). Choosing solitude: Age differences in situational and affective correlates of solitude‐seeking in midlife and older adulthood. The Journals of Gerontology, Series B: Psychological Sciences and Social Sciences. Advance online publication. https:// doi.org/10.1093/geronb/gby044
8. Ashish Vaswani, Noam Shazeer, Niki Parmar, JakobUszkoreit, Llion Jones, Aidan N Gomez, LukaszKaiser, and Illia Polosukhin. 2017. Attention is allyou need. InAdvances in Neural Information Pro-cessing Systems, pages 6000–6010
9. Koch, P. (1994). Solitude: A philosophical encounter. Chicago: Open Court.
10. Wang, Yao. “Culture and solitude : meaning and significance of being alone.” (2006).
11. Long, C. R., Seburn, M., Averill, J. R., and More, T. A. (2003). Solitude experiences: Varieties, settings, and individual differences. Personality and Social Psychology Bulletin, 29(5):578–583.
12. Knafo, D. (2012). Alone together: Solitude and the creative encounter in art and psychoanalysis. Psychoanalytic Dialogues, 22(1):54–71.
13. Coplan, R. J., Hipson, W. E., Archbell, K. A., Ooi, L. L., Baldwin, D., and Bowker, J. C. (2019a). Seeking more solitude: Conceptualization, assessment, and implications of aloneliness. Personality and Individual Differences, 148:17–26.
14. Coplan, R. J. and Bowker, J. C. (2017). “Should we be left alone?” psychological perspectives on the costs and benefits of solitude. In Cultures of Solitude: Loneliness, Limitation, and Liberation. Peter Lang Frankfurt.
15. Kiritchenko, Svetlana et al. “SOLO: A Corpus of Tweets for Examining the State of Being Alone.” LREC (2020).
16. Kiritchenko, S., Zhu, X., and Mohammad, S. M. (2014). Sentiment analysis of short informal texts. Journal of Artificial Intelligence Research, 50:723–762.
17. Calderon, Pio. “VADER Sentiment Analysis Explained | by Pio Calderon | Medium.” *Medium*, Medium, 10 Apr. 2017, https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd9.
18. Coplan, Robert J., et al. *The Handbook of Solitude*. John Wiley & Sons, 2021.
19. Courtney. “What Are Positive and Negative Emotions and Do We Need Both?” *PositivePsychology.Com*, https://www.facebook.com/positivepsychologycourses, 27 Apr. 2019, https://positivepsychology.com/positive-negative-emotions/.
20. Deepanshi. “Text Preprocessing NLP | Text Preprocessing in NLP with Python Codes.” *Analytics Vidhya*, https://www.facebook.com/AnalyticsVidhya/, 25 June 2021, https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/.
21. Dickinson, Brian, et al. “Dimensionality Reduction of Distributed Vector Word Representations and Emoticon Stemming for Sentiment Analysis.” *Journal of Data Analysis and Information Processing*, no. 04, Scientific Research Publishing, Inc., 2015, pp. 153–62. *Crossref*, doi:10.4236/jdaip.2015.34015.
22. “Google Colab.” *Google Colab*, https://colab.research.google.com/github/google-research/bert/blob/master/predicting\_movie\_reviews\_with\_bert\_on\_tf\_hub.ipynb#scrollTo=xiYrZKaHwV81. Accessed 22 Nov. 2022.
23. google-research. “GitHub - Google-Research/Bert: TensorFlow Code and Pre-Trained Models for BERT.” *GitHub*, https://github.com/google-research/bert. Accessed 22 Nov. 2022.
24. Hutto, C., and Eric Gilbert. “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text.” *Proceedings of the International AAAI Conference on Web and Social Media*, no. 1, Association for the Advancement of Artificial Intelligence (AAAI), May 2014, pp. 216–25. *Crossref*, doi:10.1609/icwsm.v8i1.14550.
25. Khalid, Samia. “BERT Explained: A Complete Guide with Theory and Tutorial | by Samia Khalid | Medium.” *Medium*, Medium, 3 Nov. 2019, https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c.
26. Lay, Jennifer C., et al. “By Myself and Liking It? Predictors of Distinct Types of Solitude Experiences in Daily Life.” *Journal of Personality*, no. 3, Wiley, Mar. 2019, pp. 633–47. *Crossref*, doi:10.1111/jopy.12421.
27. Medhat, Walaa, et al. “Sentiment Analysis Algorithms and Applications: A Survey.” *Ain Shams Engineering Journal*, no. 4, Elsevier BV, Dec. 2014, pp. 1093–113. *Crossref*, doi:10.1016/j.asej.2014.04.011.
28. Mohajon, Joydwip. “Confusion Matrix for Your Multi-Class Machine Learning Model  .” *Towards Data Science*, 29 May 2020, https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.
29. Prabhu. “Understanding NLP Word Embeddings — Text Vectorization.” *Towards Data Science*, 11 Nov. 2019, https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223#:~:text=Word%20Embeddings%20or%20Word%20vectorization,into%20numbers%20are%20called%20Vectorization.
30. “Python | Lemmatization with NLTK - GeeksforGeeks.” *GeeksforGeeks*, 6 Nov. 2018, https://www.geeksforgeeks.org/python-lemmatization-with-nltk/.
31. “Removing Stop Words with NLTK in Python - GeeksforGeeks.” *GeeksforGeeks*, 22 May 2017, https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.
32. Srinivasa-Desikan, Bhargav. *Natural Language Processing and Computational Linguistics*. Packt Publishing Ltd, 2018.
33. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
34. Lo, Siaw Ling, et al. “Multilingual Sentiment Analysis: From Formal to Informal and Scarce Resource Languages.” *Artificial Intelligence Review*, no. 4, Springer Science and Business Media LLC, Aug. 2016, pp. 499–527. *Crossref*, doi:10.1007/s10462-016-9508-4.

**APPENDIX:**