**Contents**

1. Abstract
2. Introduction
   1. Background
   2. Problem
   3. Aim
3. Literature Review
   1. Emotional Detection
   2. Dataset
   3. Sentiment Analysis
   4. Theories of Emotions
   5. Chatbot
4. Product Specification
   1. Technical Analysis
   2. Features
   3. Functional Requirements
5. Methodology

5.1 Sentimental Analysis

5.1.1 Why sentimental analysis

5.1.2 Correlation between emotions and sentiments

5.1.3 Picking the sentimental analysis tool

5.2 NLP pre-Processing

5.3 Machine Learning

1. Product Implementation
   1. Replit Hosting
   2. Creating Hosting and Coding the Bot
   3. Web Server and Pinging
   4. UI and Display
   5. Database
   6. General Emotional Review
   7. Personal Emotional Review
   8. Other Features
   9. Implemented Commands
2. Evaluation
   1. User Based Evaluation
   2. Data Based Evaluation
   3. Technical Evaluation
   4. Discussion
3. Project Management
4. Conclusion
   1. Conclusions
   2. Innovations
   3. Further Study
5. **Abstract**

Messenger application is a crucial part of our life. It is a vital part of day-to-day communication. It’s been used from important notifications and meetings to mundane conversation about the weather. Some of the notable ones are WhatsApp Messenger, Line, and Discord. With 5 billion, 500 million, and 100 million users respectively. Discord in particular is extremely popular with the younger generation as it is specifically made for gaming and youth [1]. These demographics are the most likely to experience anxiety, depression and radicalisation which is I decide to focus my project on them.[2]

The goal of this project is to create a tool to help with the mental health of the young people in discord by detecting their emotions from their messages and when necessary, sending them resources that they might need. To achieve this goal, a bot is created that can process the messages from the users in a discord server. The messages then are fed into a ML algorithm which will classify its emotion. After that it is saved into a database. At the start of each week, the bot will then evaluate their emotions and if it passes a certain threshold, the bot will send them online resources that can help them deal with their emotions.

1. **Introduction**

**2.1** **Background**

Messenger applications have significantly impacted our lives. It is currently one of the most common forms of communication, especially during the 2021 Covid Pandemic. Most of the communication in these applications are done in text. There are several popular applications that people commonly use such as WhatsApp, Facebook messenger, Telegram and Discord. Most of these are aimed for different demographics. Discord for example is aimed toward younger people [1].

Mental Health for younger people is a critical issue. It is the most vulnerable demographic with the highest percentage of members with depression, anxiety, and radicalisation.[2] A big problem is the lack of detection in these issues, and the tendencies that younger people tend to ignore the problem until it becomes severe. There is also the problem where these younger people get the wrong resource to help them deal with their issues.

The good thing is these younger people tend to be digitally literate [3]. Which means a lot of them are using the messenger app as the primary communication method. A popular messenger app is Discord which is geared towards gamers and geeks which tends to be bullied and have mental health issues [5].

Therefore, a possible solution to this problem is to monitor conversations in Discord and look for signs of mental problems. From their emotions in their day-to-day conversation. This is the idea behind this project.

**2.2 Problem**

The problem the project is tackling is detection of emotion in messenger apps. Specifically in discord. Another problem is on helping the user deal with their emotional problems, be it through giving online resources or other functions

**2.3 Aim**

The aim of the project is to create a discord bot that can read messages in discord channels, classify its emotion, and detect if the message sender is having emotional problems. When the messenger sender is deemed to have emotional problems, the bot should send them resources they can use to help themselves. The bot should also be able to provide basic help such as giving motivational quotes and chatbot to cheer up the users.

1. **Literature Review**

**3.1 Emotional detection**

Emotional detection is not exactly a new field. There have been a lot of experiments and results in these fields. A paper made by Nandwani and Verma(2021) has compiled a list of some previous research on this subject. The paper also contains a compilation of research on sentimental analysis and other things.

A notable research on machine learning was from Chaffar and Inkpen(2011)[7]. It was made using N-grams and bag of words for feature extraction and Naïve Bayes, decision tree and SVM for the machine learning algorithm. It found out that SVM works best in this case and gave an accuracy of 71%/81% depending on the database. Chaffar and Inkpen also created their own database for this purpose.

Another research done by Chowanda et al(2021) used both machine learning and deep learning. They pre-processed the text with tokenization, removal of stop words, filtering tokens based on length ,stemming ,case folding then extracting text features. Then it was trained on several algorithms from ANN to naïve bayes. The conclusion was generalised linear regression (GLM) gave the best result, with an F1 accuracy of 0.901. This paper has four different emotion categories, which are joy, sadness, fear and anger. In all algorithms, joy consistently performs best in accuracy and precision, with anger and fear switching positions depending on the algorithm. Sadness is consistently the worst performer in all algorithms. An interesting to note is in this case GLM outperformed Artificial Neural Network by a margin of ± 5%

**3.2 Data set**

To create an emotional detection machine, a dataset needs to be used. It is the dataset that we use to train and test the emotional algorithm. A lot of past projects used the International Survey of Emotional Antecedents and Reactions. Those projects include research by Adoma et al (2020), Batbaatar et al (2019) and Sailunaz and Alhajj (2019).

The Isear dataset[9] is a project made by a group of psychologists. It was led by Klaus R.scherer and Harald Wallbott. The project was made from student respondents who are asked to report a situation where they feel 7 major emotions(joy, fear ,anger, sadness, disgust , shame and guilt).This data is gained from 3000 respondents in 37 countries. As this data is made by psychologists for psychological purposes, the data seems reliable, clear and diverse enough for analysis. However this data only consists of around 7000 sentences.

Several other datasets we checked were SemEval[9] ,Emobank[10], Emotex[12], and Alm gold. However most of these either contain irrelevant data from news/stories or were datamined from twitter. There is a problem with the twitter data that most of them are datamined based on their hashtags.Upon further inspection, without proper inspection by actual humans, a lot of the data in these twitter accounts are faulty.

A thing to note is “neutral” emotion data is hard to find. Most of the past research tends to ignore neutral as an emotion, mostly because neutral does not exactly fit with most theories of emotion, including plutchik’s wheel of emotion which are the basis of almost all previous research cited in this project.

In the end this project decided to use the ISEAR dataset as our main training data, with some SemEval to extend the database. This gave the project a dataset of around 9000 sentences in four emotions, with 2200ish sentences for each emotion.

**3.3 Sentimental Analysis**

For the sentimental analysis, this project will use rule based sentiment analysis. There are several lexiconsand rules based models that can be used. The most popular being the VADER(Valence Aware Dictionary and sEntiment Reasoner)[14] analysis. With several other methods such as textBlob as other options. The project will compare these methods on the methodologies part.

Rule based sentiment analysis works by analysing a sentence, then it will check it with its trained corpus and determine the positivity ,neutrality, and negativity of a sentence. It also has compound count, which is the combined value of all three. For easier visualisation on how VADER works, look at the table below.

|  |  |
| --- | --- |
| **Sentence** | **Sentiment** |
| Hello | 'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0 |
| Hello, I | 'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0 |
| hello, i am | 'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0 |
| Hello, I am happy | 'neg': 0.0, 'neu': 0.448, 'pos': 0.552, 'compound': 0.5719 |
| Hello, I am not happy | 'neg': 0.428, 'neu': 0.572, 'pos': 0.0, 'compound': -0.4585 |

**3.4 Psychological theory**

The emotional detection is based on the Plutchik wheel of emotion[14]. The reason this was chosen is simply because most datasets and papers I found follow this theory, which meant making the ML model would be easier. While the plutchik wheel of emotion has 8 emotions, the project decided to simplify it to 4, following each emotion in the x and y axis of the wheel.

Figure 1: Plutchik wheel of emotion

Diagram

Description automatically generated

**3.5 Chatbot**

The chatbot is not the main part of this project, however as the project will also provide this service to the users. Some research are done on this part. An interesting past model on this topic is the Microsoft GPT-Dialog-2[15], a retrieval based chatbot made from reddit data. The model uses a generic transformer model and leverages a stack of masked multi-head self attention layers to train on massive web-text data.

**3.6 Discord**

Discord is a messenger application mostly used by young people. Users mostly communicate in a server usually made for a certain community(for example the UoB Computer Science Society(CSS) have a discord server), in which there are several channels in which people can discuss in which several conversation can happen at the same time. Here is an example of its display.

A screenshot of a computer

Description automatically generated with medium confidence

1. **Technical Specification**

**4.1** Technical Analysis

While neither discord bot or emotional detection on text is a new field, combining both have not been done before. So before actually designing the system, an analysis on what technical traits this bot will have to comply needs to be done.

Since discord is a messenger app, an active user can easily reach 100 messages a week. Which means the bot will be dealing with a lot of data. These have two impacts. The positive is we will be dealing with big data, which means the required accuracy for the emotional classification is lower, since its easier to get the majority of emotions correct is much easier compared than having the bot be right all the time.

Another characteristic of discord is since it’s a messenger app, its used a lot for day to day conversation. Day to day conversation unlike twitter messages tends to be neutral. So the bot must be able to categorize emotion as neutral.

On the plus side, since the project is about a discord bot, there is no need to create a new UI or backend, as both will be provided by Discord.

A discord bot needs a hosting service. There are several options, with Heroku and Replit being the popular ones. Whichever one the bot is hosted on, there’s a limited amount of space available, meaning the space complexity for the bot, the database and the models should be minimal. And the fact every process will be send to the hosting service means its must also have a good time complexity, as otherwise it would have problem processing sentences when many users are sending requests.

Finally, the fact that people type differently means making a singular evaluation for extreme emotion will be problematic. For example, a person that swears a lot will always have lower sentiment compared to a polite person. However, that does not mean the former is always less happy than the latter. Therefore, a solution must be made to evaluate each user based on metrics made specifically for that user.

Overall, these are the advantages and challenges of this project

|  |  |
| --- | --- |
| Advantage | Challenges |
| Big data | Irregularity in chatting (Typos , slank, acronyms) |
| No UI needed | Storage Limit on replit bots |
|  | The internet and Server side hosting. |
|  | Lack of reliable data |
|  | Differences between people chatting style |
|  | Neutrality |

**4.2 Features**

**Chatbot**

The bot should be able to chat with users on the server. It should be able to maintain a decent conversation. The bot should also have a second mode based on pop culture. The chatbot should be able to chat from anime dialogues

**Give quotes**

The bot should be able to give motivational quotes to motivate the users. We use actual quotes given from actual famous people, as ai generated quotes tend to give funny but not actually motivational quotes

**Emotion review**

The users will be able to get their most common emotion this day, this week and the week before.

**Emotional detection**

The bot should be able to get sentiment analysis and emotion.

**Functional requirement**

User interaction

* 1. The bot must be able to read messages from the users
  2. The bot must be able to send messages to the user
  3. The bot must be able to access discord servers
  4. The bot must be able to send message to the server
  5. The Bot must be able to send messages to user when needed

Bot message processing and emotional classification

2.1 Bot should be able to correct spellings for minor typos

2.2 The Bot must have a dictionary for slanks and acronym

2.3 Bot must be able substitute slanks and acronym with its meaning

2.4 Bot must be able to convert emoji to to text and understand its meaning

2.5 Bot must be able to get sentiment from messages

2.6 Bot must be able to get emotion from messages

2.7 Bot must be able to tag the sentences as neutral or send them for reclassification based on emotion.

Bot Database management

3.1 Bot can save emotion and sentiment to the weekly relational database

3.2 Bot must be able the important data from the weekly relational database to the permanent database

3.3 Bot will delete the weekly relational database each week

Bot – User Interaction

4.1 User must be able to get their most common emotion that week

4.2 User must be able to get their most common emotion that day

4.3 User must be able to get their most common emotion last week

4.4 User should be able get motivational quotes from the bot

4.5 User must be able to turn on chatbot

4.6 User must be able to turn off chatbot

4.7 Bot must be able to send user online resources when the bot think the user needs it

4.8 Bot must not not read messages from itself (with some exception)

ChatBot

5.1 Chatbot can respond to user

5.2 Chatbot can alternate between the two chat mode it have

5.3 Chatbot settings in a server does not affect its setting in other server

**5. Methodologies**

**5.1 Sentimental analysis**

**5.1.1 Why sentimental analysis?**

The project is using sentimental analysis due to several reasons. As stated in the challenges, the bot will need to categorize whether an emotion is neutral or others. There are several ways to solve this. The obvious one would be simply categorizing neutral as an emotion and handle it as another emotion to categorize. However, this is impractical because as stated in the literature review, finding “neutral” sentence dataset is hard, as they are surprisingly rare.

Therefore, this project chooses to opt for an alternative. Instead, the project looks into sentimental analysis. As stated in the literature review, sentimental analysis works by analysing a sentence based on a list of rules and lexicon, then it will give a rating of how positive or negative it is. Under most rule based statements of facts or standard conversation answers like “hello” , “yes” , or “thank you” will have 0 sentiment. According to Hutto and Gilbert(2014)[15] research on VADER sentiment analysis, anything between -0.25 to 0.25 is generally speaking considered neutral.

This project chose to combine normal emotional detection with sentiment analysis, the reason is because if a rule-based sentiment analysis can get the score of a sentence, then we can get its rating and from there determine if its neutral or not. This also allows us to rate how intense the emotion is in a sentence, which means it would be possible for the project to differentiate between a mild sentence (example: “I am a bit upset”) with an intense sentence (example : “I am very mad”). Under normal previous researches, both sentence will be simply categorized as angry, however under our system it will be categorized as (angry,-0.35) and (angry,-0,72) which would be useful in a big data with a lot of texts, as it means we can get not only the most common emotion but also the average sentiment of a person.

Another possible application of this emotional classification + sentimental analysis method is to act as verification for the emotional classification. By verification, it meant that for a sentence to be classified as a certain emotion, it must pass a certain sentiment score. For example, say the sentence “I like cats” is considered a happy emotion by the algorithm, however it has an sentiment of 0, in this case the machine will classify this sentence as neutral. But before this algorithm can be made, the range of the sentiment for each emotion must be found.

**5.1.2 Correlation between emotional detection and sentiment analysis**

Correlation between sentiment and emotions has long been documented.On general, negative emotions tend to have negative sentiment and positive emotions tend to have positive emotion. However for further clarification, further research is done in this section

# The emotions we are categorizing are fear, anger, joy and sadness. These are emotions taken from the Plutchik wheel of emotion[2] specifically the emotions on the second wheel on the x and y axis. The data for these emotions are taken from the ISEAR dataset. The sentiment analysis tool are done using VADER sentiment anaysis.

# The goal of this section is to prove that fear , anger and sadness are negative emotions with negative sentiment. It will be proven if sentences classified as said emotions had a sentiment that is lower than or equal to zero in majority. The reverse also needs to be proven for joy, where it must be overwhelmingly majority positive. Overwhelming in this case is defined as somewhere between 75%-90%

# This is the data we get from the experimentation

# Joy sentiment Chart :

# Chart, histogram Description automatically generated

# Sadness Sentiment Distribution Chart:

# Chart, histogram Description automatically generated

# Anger Distribution Sentiment Chart:

# Chart, histogram Description automatically generated

# Fear Chart:

# Chart, histogram Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Joy | Sadness | Anger | Fear |
| Mean | 0.48843 | -0,21913 | -0,23129 | -0,36349 |
| Median | 0.5719 | -0,3182 | -0,3612 | -0,4588 |
| standard deviation | 0.37070 | 0.48498 | 0.492808 | 0,4104 |
| Max Sentiment | 0.9802 | 0,9722 | 0,9717 | 0,8268 |
| Minimum Sentiment | -0,8567 | -0,9757 | -0,9682 | -0,9541 |
| Sentences with sentiment above 0 | 996 | 254 | 184 | 150 |
| Sentences with sentiment below 0 | 83 | 802 | 817 | 933 |
| total sentences | 1079 | 1056 | 1001 | 1083 |
| Percentage above 0 | 92,3% | 24,0% | 0,183% | 13,8% |
| Percentage below 0 | 7,69% | 75,9% | 81,6% | 86,1% |

The sentences that have exactly 0 sentiment have been removed from the data as upon manual inspection those data tend to be statements of facts or slightly emotional conversation data. Another point is the VADER paper[14] clearly categorizes 0 sentiment sentences as neutral.

# Looking at the graph, it seems the graph supports the statement where Joy is a positive sentiment while sadness, anger and fear is negative. It also categorizes anger as a negative sentiment emotion. However anger and sadness have a much higher standard deviation than the other two and also a mean closer to 0. Which means unlike Joy and Fear, the result is not overwhelmingly one sided. This also supported the fact that the percentage below zero for sadness and anger is much lower than fear or the above zero for joy.

# The result of this experiment lines up with the emotional detection of the previous methods, especially the one by Chowanda et al (2021)[8]. As stated in the literature review, that research concludes that joy is the easiest emotion to categorize, with anger following behind it then fear and finally sadness. This also lines up with a lot of previous research[7]. The reason is likely due to sadness and fear being simply harder to detect.

# The data shows correlation, but correlation does not equal causation. However, there is a logical reason why there is a correlation. Simply put, because happy sentences tend to have positive sentiment and negative sentences have negative sentiment, that is the definition of sentiment.

# Conclusion: Sentiment does have a correlation with emotional classification, as the data above shows direct correlation. However, for fear and sadness, the correlation is not as strong and therefore further caution is needed. Threshold for fear and sadness categorization should be lower than joy and anger.

# 5.1.3 Picking the sentiment analysis tool

# There are three sentiment analysis tool that we compared to each other, textblob, flair, and VADER

# Flair is the most complex method; its fancy text embeddings , NER and other features seems like a good candidate, however upon testing it took several seconds for flair to analyse a sentence, and therefore eliminated.

# Textblob and VADER pretty much use the same method, which is explained in the literature review. VADER is specifically geared towards social media. And because of that we picked Vader analysis as our sentiment analysis tool.

# 5.2 NLP pre-processing

# There are several pre-processing methods that any text passed before emotional detection and sentiment analysis. Here are the methods and the examples.

# Message example : OMG i really liek cats! But i can't touch them. ☹

|  |  |  |
| --- | --- | --- |
| 1 | Original Sentence | OMG i really liek cats! But i can't touch them. ☹ |
| 2 | Tokenisation | [OMG, I, really, liek ,cats!, But, i ,can’t, touch, them., ☹] |
| 3 | casefold | [omg, I, really, liek ,cats!, but, i ,can’t, touch, them., ☹] |
| 4 | Translate apostrophes | [omg, I, really, liek ,cats!, but, i ,cannot, touch, them., ☹] |
| 5 | Slank and Acronym handling | [oh my god, I, really, liek ,cats!, but, i ,cannot, touch, them., ☹] |
| 6 | Spell Checker | [Oh my god, I, really, like ,cats!, but, i ,cannot, touch, them., ☹] |
| 7 | Emojis | [oh my god, I, really, like ,cats!, but, i ,cannot, touch, them., happy ,face] |
| 8 | Lemmatization | [oh, my, god, I, really, like, cat !, but, i ,can, not, touch, them ., happy ,face], |

# Tokenization

# Tokenizing is an extremely important part of NLP. A computer cannot just magically read a sentence and understand its meaning. Therefore we must split a sentence into a set of words. This way, assuming the bot can take the words as a stand alone string which can be transformed into a matrix later.

# CaseFolding

# The reason casefolding is needed is because without casefolding the word “Hello” and “hello” will be considered as two different words. We don’t want that as that might create some inconsistencies. Especially since our database is small and its extremely likely that there are cases where some vital word like “Happy” gets capitalised only once in the database.

# Translate Apostrophes

# The reasoning for this part is the same as casefolding, it’s to prevent inconsistencies in the text. For example, the words ``can't” and “cannot” is the same word, but without this it would be considered different words. The way the bot does it is by using regex to find apostrophes and check if its in a replaceable context

# Slank and Acronym

# This part is done as a translation service for modern words/acronyms that does not appear in the database. This part is done by adding a dictionary to the database. Then it will check the list of words it received from the tokenisation If the word is not a word ( defined by NLTK.word[17]) then check if the word is in the acronym list. If the word is in the acronym/slank list then it will be replaced with its meaning or more common equivalent

# Spell Checker

# The spell checker is made to correct spelling. After complaints from users it was deactivated and further review it’s deactivated for now. The way the spell checker works is by checking the word and finding the nearest neighbour using word distances as defined by NLTK library[17].This is a fairly standard spell checker that is commonly used.

# Emojis

# Emojis in discord have names with a certain format. It starts with ‘:’ then it will have a string with ‘\_’ as space and it will end with another ‘:’. Example : ( “:smile\_face:” and “:gun:” ) . It is then passed into the bot as UTF-8 code.

# To translate emoji into words, the bot can read the UTF-8 code using the existing library and then translate it to string. Then with regex the bot can convert the words into string which are then tokenized.

# Lemmatization

# Another step the bot takes is lemmatization. Most of the previous research chooses to do steming instead. Personally I don’t understand the logic behind it , as it's never explained why they choose steming over lemmatization . To choose between lemmatization and stemming, a test was run. The test was run with a naïve bayes algorithm, with 70/30 split. The test was done 100 times with different splits each time. For more details check the Lem vs Stem Ipynb.

# The result was a comparable accuracy for Lemmatization and Stemming. with the scores are:

|  |  |
| --- | --- |
| normal | 0.7108837798 |
| Lem | 0.714755115 |
| stem | 0.7149495887 |

# So in this case we decide to use Lemmatization, as both have comparable results. However an argument for stemming is also valid.

# The lemmatization is not done for sentiment analysis, it's only done for emotional detection. The reason is because upon testing VADER does not need lemmatization as they have their own algorithm.

# After the bot has done all these, the data is then fed into the sentimental analysis and the emotion detection. This process is done both in the training phase and in the deployment phase(when discord messages are received)

# Below is a diagram on how the NLP works

# Diagram, engineering drawing Description automatically generated

**5.3 Machine Learning**

Emotional Classification from text is at its core a multi-class machine learning problem. To solve this problem we need to try out existing solutions to multi class machine learning problems and see which one of them suits our needs the most. Keep in mind that we have several factors to consider before picking the time(as in how fast it processes a sentence) , space(how big is the model and how long does it take to train one) and accuracy. The space factor is insignificant until we get into deep learning.

The first few steps the bot takes is the same for all algorithms. At first, the bot will take the data from the NLP and vectorize it using countvectorizer, the accents will be stripped to ascii. And the features are limited to 100000. In this case the features will be the words from the training. An interesting setting is the Ngram setting. In this case ngram means how many words are considered to be a single feature. For example, the word “hello there” will be considered two different features in 1 word ngram, but as one feature in 2 word ngram. Logically(and backed by result) 2 word ngram was never an option. There is however the option of 1-2 word ngram, which means in case of “hello there '' there will be 3 features : hello, there and hello there.

The result from testing 1-2 ngrams gave slightly better results at the expense of 6 times the amount of features. Considering how small the improvement is, it is decided to stick with 1-1 ngrams

|  |  |  |
| --- | --- | --- |
| **Ngram** | **Features** | **Accuracy** |
| (1,1) | 13132 | 0.7819347942276857 |
| (1,2) | 78771 | 0.7844863709246392 |
| (1,3) | 176605 | 0.7840726884019241 |
| (2,2) | 65639 | 0.615713522180652 |

After the Ngrams, the result is then run through Term frequency-inverse document frequency. The effect is instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus. In this case it will reduce the impact of common words like “I”, “who” or “we” which can appear in any emotion.

And to finally pick the method, We are evaluating a few processes here, namely multinominal naïve bayes, SVM, Logistic regression, decision tree and k nearest neighbour. As a bonus, we also train an LSTM+CNN model and compare it to the classic machine learning techniques. The parameters in the test are the best parameters we can get after trial and error.

The algorithm we use is the same

explain how the algorithm works

Result:

|  |  |
| --- | --- |
|  | Test 1 F1 Accuracy |
| Multinomial Naïve bayes | 0.7589524318546232 |
| SVM | 0.7733832175307322 |
| Logistic Regression | 0.7530732228754676 |
| decision tree | 0.6440406199893105 |
| K nearest neighbour | 0.6098343132014965 |
| LSTM + CNN | 0.575 |

SVM gives the best result, with Naive Bayes and Logistic regression closely behind. The SVM uses. The logistic regression uses C = C=1e10 and n\_jobs=1 which seems to work best. Decision tree uses entropy as criterion and best as a splitter with the minimum sample split as 2. The K nearest neighbour N is 7. The LSTM is trained on 20 epoch, 128 batches. One thing to note is while SVM gives the best result, both logistic regression and Naïve Bayes gave a more balanced confusion matrix.

Naïve bayes confusion matrix SVM confusion matrix

Chart, waterfall chart

Description automatically generatedChart, waterfall chart

Description automatically generated

We will keep testing for the top 3 algorithms from the graph which are Naïve Bayes ,SVM and Logistic Regression. But before that, a summary on these three algorithms.

Naïve Bayes is predictive modelling algorithm based on the bayes probability theorem. Which is basically uses conditional probability. In this case it will use the features (words) to calculate the probability of a sentence being a certain emotion.

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A divider between the group is found, then the data is transformed so the divider can be a drawn as a hyperlane. Then using this divider, SVM can predict the new group of the next sentence.

Logistic regression is technically a statistic technique. It uses logistic functions to model conditional probability.

Test 1 and 2 is tested on original ISEAR +Semeval + some discord data dataset, The difference is on test and train set. Test 3 is done with Isear +Semeval pre-balancing , test 4 is done with pure ISEAR dataset and test 5 is done with Semeval.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test 1 Accuracy | Test 2 Accuracy | Test 3 Accuracy | Test 4 Accuracy | Test 5 Accuracy |
| Multinomial Naïve bayes | 0.758 | 0.762 | 0.741 | 0.731 | 0.702 |
| Svm | 0.773 | 0.786 | 0.774 | 0.744 | 0.721 |
| Logistic Regression | 0.753 | 0.762 | 0.722 | 0.701 | 0.681 |

From what we can see, SVM consistently performs better than its competitor in various databases. We need to especially highlight test 1 and 2 since the extended corpus will be our training set for the classification.

And because of that this bot is made using SVM.

**6. Product Implementation**

6.1 **How the components work together in emotional detection**

This section purpose is to explain on how the three methods from the methodologies work together. Here is the diagram of the combined method

Diagram

Description automatically generated

So, when the bot receives a message from the discord, then the message is then processed with NLP methods from part 5. Then the text is passed into the emotion classification and sentiment, which means the output is emotion and sentiment

As part 5 has concluded, there’s a correlation between emotion and sentiment. Therefore, in this part both systems are combined to create a more accurate system. First the system will check if the emotion is considered a sad/negative emotion. In this case the negative emotions are angry,sadness and fear.

Angry,Sadness and Fear have different sentiment thresholds. Originally it was -0.25 for fear, -0.15 for anger and sadness. However, feedback from the users resulted in a decrease in anger threshold, moving it to -0.20 for anger. When a sentence is categorized as having any negative emotion and does not pass the threshold they are categorized as neutral.

If the sentence is considered joyful but its sentiment is below 0.25, then it is a false positive emotion. In this case the text is reprocessed in a secondary emotion detection model. The difference between the primary model and the secondary one is simply that the secondary emotion does not have “Joy” emotion.

Then if the emotion is joy and the sentiment is between -0.25 to 0.25 then it is categorized as neutral. After that the emotion and the sentiment are passed into the database.

The reason the bot has a secondary model for joy but not for the negative emotion is based on the result from section 5.1.2. In section 5.1.2 , sentences where the emotion of a sentence is negative and the sentence also has negative sentiment only makes up between 75%-86% of the test, while situations where the sentence emotion is joy and the sentiment is positive correspond to 96% of the total. From this data, it is concluded that joy correlation with its sentiment is almost always correct. And therefore, it is almost certain that in conditions where joy is the emotion, but the sentiment is negative is an error. On the other hand, for the negative emotions, the chances of the emotion being negative and the sentiment being positive and still be correct ranges between 14%-25% so a switch into joy is uncertain and therefore the best solution is just to be neutral.

In the project this part is done in Processing.py, while the emotional detection is done in emoteProcessing.py

**6.2 Creating the bot.**

Creating a discord bot is easy and the steps to do it have been well documented [18]. A creator simply needs to follow the process in [discordapp.com/developers/applications/me](https://discordapp.com/developers/applications/me). And choose the permissions that the bot needs. In this project, the bot will need access to the text channel and access to send messages in the channels

Here are our permissions with some unneeded permission for future updates

A screenshot of a computer

Description automatically generated with medium confidence

**6.3 Hosting the bot**

While theoretically a discord bot can be hosted locally, its much more convenient to host it somewhere else. Therefore, the bot was hosted on replit. There are some alternatives hosting services such as Heroku and AWS. This project is made in replit because the creator is most familiar with replit. Creating and hosting a discord bot can be done by going to replit.com, creating a new project and then adding a python file. In the python file, import the discord library and create a discord client. Call the discord client with the bot token, then run the replit. When the replit is run, the discord bot should be online and receiving requests.

**6.4 Coding the Bot**

The bot was coded in python. A discord bot is an asynchronous program, meaning its triggered when events happen. The events that trigger the bot are messages from discord servers where the bot is present. The way the bot works is :

1. Replit watches discord servers where the bot is present

2. If a message is sent, then it's passed to the bot

3. The bot will check if the message is one of its many commands, if yes then do the action required for that code

4. If the message is not a command, then run the message in the NLP/Sentiment/Emotion Detection part and get its emotion and sentiment. Then send the result to the database.

This part is coded in main.py

**6.5** **Web Server and UI**

To keep the bot running even when the host’s device is off, a web server is created. With a webserver, the bot will keep running for one hour. After one hour, if replit does not receive any request from the server, replit will shut down the bot. To bypass this we are using an online tool called uptimerobot. It works by constantly pinging a service, in this case the bot every certain interval. With this the bot will not go offline unless it is turned off.

This part of the code is done in keep\_alive.py

Here is the setting for the uptimerobot :

Graphical user interface, application

Description automatically generated

**6.6 UI**

As previously stated, one of the main conveniences in this project is there is no ui design needed. The bot simply appears in discord. For reference here is an example of the bot display in discord :

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**6.7 Databases**

There are two databases used in the bot. A relational database for the conversation data and a key value structure database. The reason is because replit recommends users to use the provided key value structure database, as it is made to work with replit. However it is not possible to save the hundreds of messages from the thousands of users with all of the value that the bot needs to store. To store that, the bot will need a relational database.

To store user messages, a sqlite db is created. It uses the user (as in the people who send the messages) id as the table name. It stores several values such as message number index, the emotion, the string of the message, the intensity and the date of the message.

Graphical user interface, text, application

Description automatically generated

Using the data from this database, users can request their most common emotion of the day and week. These are done by using an SQL query and matching the date. These processes are done in databaseProcessing.py

Saving these data is space intensive, and for that reason every week the relational database is dropped. Before the database is dropped, important values are calculated and moved to the key value structure database which only saves 3 values for every user. The details for the saved values will be in the Personal Evaluation part below.

The key value structure database works by using serverId and userId as keys. To access it, the bot only needs to request the value from a key. serverId and userId are easily distinguishable because in the processing the bot added the string “id” to userId and left serverId as a pure number string. In serverId is saved the server bot settings such as whether the chatbot feature is online in the server. In the userId the server saves their average sentiment, the number of weeks the bot has been analysing them and their most common emotion the previous week.

Note : If the bot have not found any emotion in a user database, the $myEmotionToday function won’t send anything.

**6.8 General Emotional Review**

This section will explore weekly emotional review in situations where the previous user does not have any existing data.

Once every seven days, the data from the SQL database will be evaluated. First the evaluation will save all the data names, as they are the active users in that week. Then the SQL will be converted into CSV and then split into lists. The list is based on the userId of the sender.

The evaluation then will calculate the average sentiment and get the most common emotion for that user. This process is then replicated for all users. The end result should be a list of users, their number of messages, their average sentiment and emotion. Example ([id1, 102,Happy,0.00324],[id2,84,sadness,-0.0012]) they are then split further into two lists based on their number of messages. If a user sends more than 101 messages, then their data is categorized as relevant data, if a user has less than 101 messages, then their data is considered as irrelevant. There are several reasons for this categorization, to understand it, look at the data below

Here is a dataset from 17 march 2022 after splitting into relevant and irrelevant datas

Relevant data

|  |  |
| --- | --- |
| id | sentiment |
| id305188118666018817 | -0.006257646277 |
| id401371902213488651 | 0.002296283784 |
| id432610292342587392 | 0.002212284569 |
| id482906266654736404 | 0.002817197452 |
| id828277316517888011 | -0.004873417722 |
| id756403906543747122 | -0.00669025 |
| id737127811785031711 | 0.01023015267 |
| id340495377017077761 | 0.0100939759 |

Irrelevant data with bots removed

|  |  |
| --- | --- |
| id | Sentiment |
| id693205371652800545 | -0.06698333333 |
| id433279630762442753 | 0 |
| id347211683510353920 | 0 |
| id817972348262809630 | 0.8212 |
| id488375454881808384 | -0.1158666667 |
| id310872670298439680 | -0.0903 |
| id764446149960269854 | 0.07025 |
| id751104538898858015 | -0.008887012987 |
| id664842665124167681 | 0.006741860465 |
| id408785106942164992 | 0.4574 |
| id348050913711095811 | 0 |
| id452792458049224715 | -0.001745454545 |
| id893961991390121984 | 0.02968727273 |
| id241831656183300097 | 0 |

From this data, the difference between the relevant data and irrelevant data becomes obvious. The relevant data have an extremely small range, with the difference between the max data with min data being 0. 01645. The standard deviation is 0.006763777 and the mean is 0.001228572548. Because the range for all sentence sentiment is between -1 to 1, the difference in the relevant data ended up as 0.8% of the total range, this makes sense as most conversations are neutral and are consistent with the weekly report. The mean is also near 0, which is what is supposed to happen.

Meanwhile the irrelevant data range is all over the place, with a lot of value being much higher than 0 or exactly 0. This is not supposed to happen, because most conversation in a messenger application is neutral. The range for the irrelevant data this week is 0.93 or 46.5% of the total range, with its average being 0.07867833326 and its standard deviation being 0.2525150558. None of these data make sense, the distribution is too big and the average is too high.

From these data it can be concluded that counting average message sentiment before a certain point is not wise. However, the exact cut off point needs to be determined. In this scenario the cutoff point will be when if a message is sent, the average sentiment change must be within ± 1 assuming the average is 0. Therefore, the number of sentences should cover all 4 of these conditions

| ,

, and

The N are 99, 99,101 and 101. And therefore, to satisfy all conditions, the bot uses 101 as the threshold.

There are two effects from the relevant/irrelevant data split. Firstly, it meant the standard deviation and the mean of the data will be more accurate with the removal of the invalid data. Two, it meant that for a week where the user is inactive/only slightly active, their data won’t be saved and therefore it won’t ruin their statistics.

After the split, the bot analyses both the relevant and irrelevant data. In the analysis part, the bot will get the average mean and standard deviation of the relevant data and compare them to the average sentiment of the users. If the user average sentiment is outside the standard deviation ± threshold, then that user is categorized as having an extreme emotion that week.

Text

Description automatically generated

This is the result of the review. These are the users that are categorized here are the ones with extreme emotion. Then if a user is in the relevant data list and is in this list, the bot will send them messages like this.

Text

Description automatically generated

After this important data are saved into the database. For user in the relevant category, the data saved are the number of weeks that user have been active(in this case it should be 1 since we don’t have prior data in this case) , the average sentiment and their most common emotion that week

Text

Description automatically generated

For the irrelevant data, we do not increment the active weeks, so it will start at 0. The sentiment is also not updated. However we do update the emotion, So the users can get their emotion last week in $myEmotionLastWeek command.

After the database is moved, the SQL database is then dropped and renewed.

This section is coded in weeklyProcessing.py

**6.9 Personal Emotional Review**

The personal review is done like the general emotional review with several key differences. One, this will only happen when the user data exists in the database.

Two, instead of comparing the user’s weekly average sentiment with the average sentiment of all users, the bot instead compares the user’s weekly average sentiment with the user accumulated average ± weekly standard deviation. This means that if a user constantly has negative sentiment because of their talking pattern, they won’t be constantly considered as having a negative emotion. While cases where said the user always have a negative emotion can sometimes happen, according to research [19], those cases are rare, as mood swings only last between hours to days.

The third difference is the database. While the irrelevant datas is updates the database the same way as in the general method(just replacing emotion last week), in the relevant users the weeks active is incremented. The average sentiment is also changed, with the formula being and the recorded emotion is also changed.

**Other features**

There are several other features implemented in this bot. This section will discuss them briefly. Those are chatbot, jokes, and quotes

**Chatbot**

There are two chatbots available for the bot, those are GPT-Dialog[21] and Anime ChatBot[20]. Gpt dialog was made by Microsoft by training a chatbot from conversation data. The anime Chatbot was made for this project using data gathered from anime conversations. Then several characters are picked as the baseline for the bot personality, with dialogue from other characters being set as context. The hope is to get the bot to understand context and able to reply accordingly. Both chatbots are retrieval based, which means the responses are limited to trained messages. The anime chatbot is then trained with CNN.

Both chatbots are then saved to [huggingface.co](https://huggingface.co) . Hugginface.co is a website designed to host text-based bots. The bots are hosted there because they don’t fit in the replit database. The chatbot is then connected to the emotion bot. When the chatbot is online, all messages in the server will be sent to the emotion bot which will forward it to hugging face, then hugging face gets the reply from the chatbot which is then send back to the server.

Chatbot display on the hugging face website :

A screenshot of a computer

Description automatically generated

Chatbot in the discord server :

Graphical user interface, text, application

Description automatically generated

The setting for the chatbot is saved under the serverId key. It has 3 possible values : 0,1,2. 0 means the chatbot is Off, 1 the chatbot is on(normal mode) and 2 means the chatbot is in anime mode.

This section is coded in chatbot.py.

**Giving Motivation Quotes**

This part is made by requesting API from <https://zenquotes.io/api/random>. Then the website will give a motivational quote which is then forwarded to discord.

**Jokes**

This part is made by getting a list of jokes. When requested, the bot will send jokes from the list.

User Command :

|  |  |
| --- | --- |
| Command | Effect |
| $myEmotionToday | Get user’s most common emotion that day that is not neutral |
| $myEmotionThisWeek | Get user’s most common emotion that week that is not neutral |
| $myEmotionLastWeek | Get user’s most common emotion the previous week that is not neutral |
| $gibQuote | Get a random motivational quote |
| $giveStats | Give user data from the key value database |
| $chatBotOn | Turn on chat bot |
| $chatBotOff | Turn off chat bot |
| $chatBotAnime | Turn on anime chat bot |
| $joke | Send a joke |

Debugging command :

|  |  |
| --- | --- |
| Command | Effect |
| $reviewTime!124 | Triggered by the both, used to start weekly evaluation |
| $unitTesting | Trigger unit testing, for testing purpose only |
| $databaseEvaluation | Manually trigger evaluation |
| $getChannelId | Get discord channel Id |
| $sendMsg | Get user Id and send message to the user |

**7) Evaluation**

To evaluate the effectiveness of the project, several methods are taken. The first method would be comparing the bot analysis of people’s emotion with their actual emotion. Two would be testing some sentences with the combined algorithm. Method three would be technical evaluation.

**7.1) User Based Evaluation.**

In this section, an evaluation of the bot will be done using the result from the weekly review of 24-3-2022. The reason this date is picked is because at this point most of the users had at least more than 1 active week. The result of the bot for this week is then compared to the user's actual feeling gained from an interview:

|  |  |  |
| --- | --- | --- |
| User Id | Predicted emotion | Actual emotion |
| id305188118666018817 | Joy | Joy |
| id488375454881808384' | Neutral | Neutral |
| id753545054592827392 | Joy | Joy |
| id756403906543747122 | Anger | Anger |
| id777410951019888660' | Sadness | Neutral |
| id488375454881808384 | Neutral | Neutral |
| id693205371652800545 | Joy | Joy |

The result gave back a 85% percent accuracy. However, the dataset is extremely small and therefore this can’t be used as a definite proof that the bot works. Getting more weekly data is problematic, as it means interviews for the users. Because the users are not provided any monetary benefit here, they refuse to do another interview.

In the same interview, we also gained daily review data. In which several days where the users are active is picked then analysed. In this part we will compare this data with the interview result. For the code, see the evaluation2.ipynb file.

Here is how the daily review looks like when combined :

Graphical user interface, application, table

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

The daily review ended up with 75% accuracy. This is expected, as since the data is smaller, the accuracy is expected to be lower. The most common mistake is the bot categorize the user as having a negative emotion when the user feeling is neutral. This corresponds with our weekly data where the only mistake was sadness-neutral.

The evidence here is not conclusive, however from the gathered data it seems that the bot works exactly as expected.

**7.2) Data Based Evaluation**

In this section, several users provide several testing sentences(totaling at 140 messages) then the sentences are analysed by the bot. Full testing is done in evaluation2.ipynb.

The result provides a F1 accuracy of 69.2%. The lower accuracy compared to the result from 5.3 in methodologies is expected. This is because there are five emotion categories now with the addition of neutral and because actual data is more varied than ISEAR+SemEval.

Below is the confusion matrix of the sentences.

Graphical user interface, application, Teams

Description automatically generated

A thing to note is once again most of the mistake happens in neutrality. In this case the bot predicts neutral too often, especially for anger sentences. Misclassification from a non neutral emotion to another non neutral emotion only happens occasionally.

**7.3) Technical evaluation**

This section will evaluate if the bot works as intended and if its passes the functional requirements.

|  |  |  |
| --- | --- | --- |
| No | Expected Outcome | Actual Outcome |
| Bot-Discord Functionality | | |
| 1 | The Bot can read message from user | Pass |
| 2 | The Bot can receive commands from user | Pass |
| 3 | The Bot can send messages to the server | Pass |
| 4 | The Bot can send messages to user when needed | Pass |
| Bot message processing and emotional classification | | |
| 5 | Bot can substitute slanks and acronym with its meaning | Pass |
| 6 | Bot can correct spellings for minor typos | Failed |
| 7 | Bot can convert emoji to to text and understand its meaning | Pass |
| 8 | Bot can get sentiment from messages | Pass |
| 9 | Bot can get emotion from messages | Pass |
| 10 | Bot can tag the sentences as neutral or send them for reclassification based on emotion. | Pass |
| Bot Database management | | |
| 11 | Bot can save emotion and sentiment to the weekly relational database | Pass |
| 12 | Bot can save the important data from the weekly relational database to the permanent database | Pass |
| 13 | Bot will delete the weekly relational database each week | Pass |
| Bot – User Interaction | | |
| 14 | User can get their most common emotion that week | Pass |
| 15 | User can get their most common emotion that day | Pass |
| 16 | User can get their most common emotion last week | Pass |
| 17 | User can get motivational quotes from the bot | Pass |
| 18 | User can turn on chatBot | Pass |
| 19 | User can turn off chatbot | Pass |
| 20 | Bot can send user online resources when the bot think the user needs it | Pass |
| 21 | Bot does not read messages from itself (with some exception) | Pass |
| ChatBot | | |
| 22 | Chatbot can respond to user | Pass |
| 23 | Chatbot can alternate between the two chat mode it has | Pass |
| 24 | Chatbot settings in a server does not affect its setting in other server | Pass |

The bot fulfilled almost all of its intended function. And it is robust and reliable, as in the last 30 days, the bot is on 99.98% of the time. The 0.02% missing downtime is mostly because the bot was used for testing, making its uptime virtually 100%.

Graphical user interface, application

Description automatically generated

**7.4) Discussion**

In this section the overall result from the data will be discussed, it will also include several feedbacks from users of this bot and discuss whether the challenges of this project have been solved.

Overall while individually the sections are not strong enough to prove that this application works perfectly, both the user/interview-based evaluation and the sentence based review have several trends.

First, the more data the bot gets to analyse, the more accurate it is. This is observable, as on sentence-based analysis it scored a low 69% accuracy. The daily accuracy is a bit higher at 75% and the weekly accuracy goes as high as 85%. This is exactly what is expected from the start. In the system analysis, it is clearly stated that having lots of data to analyse is one of the bot strongest points.

Second point is that most mistakes are made on miss categorization on neutral, be it predicting neutrality when it's not or vice versa. This is hard to fix, as the difference between emotions to neutral is much subtler than the difference between emotions to other emotions. This is proven by the fact that in 7.2 miss-categorizations from a non-neutral emotion to a different non neutral emotion are rare compared to non-neutral to neutral. And it's not a fatal problem, as in the end, the algorithm does not count neutral emotion for the analysis, it only counts the sentiment and the most common non neutral emotion.

From a technical perspective, this bot works perfectly. All planned features are implemented, and the bot is always online. In 1 and a half of testing, the bot has never had any server or feature problem whatsoever.

In the end, this project managed to solve all of its challenges from the project management. To recap, Here are the previous table of challenges and how the project solved it :

|  |  |
| --- | --- |
| Challenges | Solution |
| Irregularity in chatting (Typos , slank, acronyms) | NLP and dictionaries |
| Storage Limit on replit bots | Lightweight models, external hosting for chatbots |
| The internet and Server side hosting. | Lightweight models |
| Lack of reliable data | Combining sentiment analysis with emotional detection, and using that to bypass the lack of neutral data |
| Differences between people chatting style | Personalised evaluation |
| Neutrality | Sentiment analysis |

**8) Time Management and Planning**

The project plan is available in several PowerPoints in the files. To recap. In the first two weeks, the creator is stuck in self isolation with barely any internet. The project starts properly at the third week. The original proposal was creating a “emotional detection chatbot to combat extremism and loneliness” Then the next three weeks was spent doing literature review and data mining for the chatbot.

Then the inspection happened, and the project took a different direction. Week 8 was then used to reconfigure the aim of the project. It is then decided that the chatbot part is not that important, after all people use discord to chat with others, not with a bot.

The new plan goes as follows.

Week 9: Research in sentiment analysis, find the sentimental analysis tool to use

Week 10: NLP processing for emotional detection

Week 11: Run sentimental analysis on ISEAR dataset, Work in emotion detection

Week 12: Finish Emotion detection, get it working reliably at least

Week 13: Finish Chatbot

From week 9 to week 13, the project goes exactly as planned. By this time the bot has every tool it needs, and the implementation can begin. However, it’s now January. In this phase the time management are clearly documented in Gitlab

The project was paused in the first two weeks of January due to exams. There are 2 pushes on 10th and 11th of January, but those are just frameworks for the eventual implementation.

24th January: Emotional detection and Sentiment analysis is implemented to the discord bot.

26th January: SQL database is implemented

30th January: More tinkering with the database, added several comments and values to save.

1st February: Created a discord server for testing purposes, invited several people to act as users

3rd February: Removed 3 emotions from the emotional classification , leaving it with inly joy, sadness, anger, fear and neutral. Also fixed some bugs in the database

8th February: Researched the exact correlation between emotions and sentiment. Used the result to recalibrate the combined emotional classification-sentimental analysis method.

15th February: Got the webserver to work, Combined ISEAR dataset with SemEval

16th February: Research on more ML algorithms. Changed the method from Naïve Bayes to SVM, discord bot can now send private messages to people

21st February: Emoji Processing

22th February: Implementation of Spell Checker and secondary model for negation

24th February: Added acronym detection and translation, apostrophes correction

1st March: Chatbot-discord Bot integration

3rd March: actually, get the chatbot to work

8th March: Start implementing general weekly evaluation

10th March: Finished the Key-Value database

11st March: Personal weekly evaluation now works; discord bot will now send messages to users after weekly evaluation

23rd March: Start making test cases

25th March: User interview and comparing it with weekly and daily data review

After the 25th of march, the project work is mostly report based. While the project might seem to be mostly done in January-March, this is not true. A lot of the January – March codes are simply implementation from the research and codes done in the previous year. Overall, the work is evenly spread except for the first 5 weeks. This timeline is well documented and can be backed by gitlab, Microsoft teams and past project plan documents.

**9) Final Words**

**9.1 Conclusion**

Overall, this is a complex subject. The project is divided into two parts, one being the theoretical problem on how to solve the challenges provided by messenger based emotional detection bots, and the second one is its implantation. Both are solved with solutions implanted in this project. While not perfect, the bot accomplishes what it needs to do to a great extent.

**9.2 Limitations**

The bot has some problems, however. The math and algorithm for determining if someone is considered having an extreme emotion don’t have a theoretical base outside of statistics. This is because there is no paper on what’s considered to be a healthy sentiment score for a human. As such, the bot chooses to compare it with their peers. If proper research is done on this subject, then the algorithm should be updated.

Another flaw in the system is the fact that the bot cannot detect the more subtle parts of language, for example sarcasm, double meaning words, idioms etc. While the bot can understand acronyms and slangs, making a sarcasm detector is not as easy, especially since even people can fail on detecting that in real life. This problem is something that haunts previous research in text-based emotion detection, and it's not the aim of the research to solve that problem anyway.

The chatbot part of the bot needs some work. As while it rarely answers something totally unrelated, it still is clearly a bot. There have been advancement on the chatbot technology recently that almost makes them human [16]. However, the model is too space and time intensive to be applied to the chatbot.

**9.3 Innovations**

This project brings some novel innovation. This is one of the first (if not the first) emotional detection bot in discord and messenger apps in general. It also combines sentimental analysis with emotional detection. This is new compared to the purely ML/Deep learning method previous research did. And with the problem of relatively neutral messages dataset, this method is arguably better in situation where neutral messages are the norm.

**9.4 Future Work**

Future work can look into fixing the flaws in this project, namely actual research on what is supposed to be the average sentiment and by how much it can deviate. Research also needs to be done to clearly detect sarcasm and idioms and translate them for the bot.

Total word without abstract and reference : 9700 words

Citations. I’ll make the citation properly later

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7364393/[3]

<https://discord.com/company(1)>

[https://www.centreformentalhealth.org.uk/sites/default/files/2018-09/CentreforMentalHealth\_MissedOpportunities\_16-25years.pdf[2](https://www.centreformentalhealth.org.uk/sites/default/files/2018-09/CentreforMentalHealth_MissedOpportunities_16-25years.pdf%5b2)]

[https://files.eric.ed.gov/fulltext/EJ824777.pdf[4](digital](https://files.eric.ed.gov/fulltext/EJ824777.pdf%5b4%5d(digital) literacy)

[file:///C:/Users/Imperator/Downloads/ijerph-18-01930-v2.pdf[5](about:blank)]

[https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402961/[6](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402961/%5b6)]

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402961/#CR31>[7]

Chowanda A, Sutoyo R, Tanachutiwat S, et al. Exploring text-based emotions recognition machine learning techniques on social media conversation. *Procedia Comput Sci.*2021;179:821–828.[8]

[**https://www.unige.ch/cisa/research/materials-and-online-research/research-material/[9**](https://www.unige.ch/cisa/research/materials-and-online-research/research-material/%5b9)**]**

Ma X, Zeng J, Peng L, Fortino G, Zhang Y. Modeling multi-aspects within one opinionated sentence simultaneously for aspect-level sentiment analysis. *Future Gener Comput Syst.*2019;93:304–311[10]

Buechel S, Hahn U (2017) Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In: Proceedings of the 15th conference of the european chapter of the association for computational linguistics: volume 2, Short Papers, pp 578–585[11]

Hasan M, Rundensteiner E, Agu E (2014) Emotex: detecting emotions in twitter messages. In: 2014 ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference. Stanford University, Academy of Science and Engineering (ASE), USA, ASE, pp 1–10 [12]

Agrawal A, An A (2012) Unsupervised emotion detection from text using semantic and syntactic relations. In: 2012 IEEE/WIC/ACM international conferences on web intelligence and intelligent agent technology, pp 346–353. 10.1109/WI-IAT.2012.170.[13]

**Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.**.[14]

[**https://arxiv.org/abs/1911.00536[15**](https://arxiv.org/abs/1911.00536%5b15)**]**

[**https://github.com/lukalabs/replika-research[16**](https://github.com/lukalabs/replika-research%5b16)**]**

[**https://www.nltk.org[17**](about:blank)**]**

[**https://www.digitaltrends.com/gaming/how-to-make-a-discord-bot/[18**](https://www.digitaltrends.com/gaming/how-to-make-a-discord-bot/%5b18)**]**

[**https://www.verywellmind.com/what-are-mood-swings-1067178#:~:text=These%20mood%20shifts%20are%20typically,to%20abandonment%20(real%20or%20imagined)[19**](https://www.verywellmind.com/what-are-mood-swings-1067178#:~:text=These%20mood%20shifts%20are%20typically,to%20abandonment%20(real%20or%20imagined)%5B19)**]**

[**https://huggingface.co/grayson124/chatbotwaifu[20**](https://huggingface.co/grayson124/chatbotwaifu%5b20)**]**

**https://huggingface.co/microsoft/DialoGPT-medium[21]**