

Data Science Applications

**Final project Report**

**“Emotion based chat bot​”**

**Group 6**

**Prepared by:**

Esraa Badawi

Esraa El-kot

Salma Sultan

Sondos Ali

**Under supervision of**

Prof. Arya Rahgozar

Contents

[Overview 3](#_Toc110367149)

[Problem Formulation 3](#_Toc110367150)

[System Overview 3](#_Toc110367151)

[Conversation Flow 3](#_Toc110367152)

[System Components 4](#_Toc110367153)

[Datasets 4](#_Toc110367154)

[Emotions dataset 4](#_Toc110367155)

[Quotes dataset 4](#_Toc110367156)

[Bot responses dataset 4](#_Toc110367157)

[Components Implementation 4](#_Toc110367158)

[Emotions classifier 4](#_Toc110367159)

[Loading dataset 4](#_Toc110367160)

[Data visualization 5](#_Toc110367161)

[Data preparation 9](#_Toc110367162)

[Classification models 9](#_Toc110367163)

[Champion model analysis 11](#_Toc110367164)

[Error analysis 12](#_Toc110367165)

[Quotes clustering model 15](#_Toc110367166)

[Loading dataset 15](#_Toc110367167)

[Data visualization 15](#_Toc110367168)

[Data preparation 16](#_Toc110367169)

[Feature Engineering 17](#_Toc110367170)

[Modelling using K-Means 17](#_Toc110367171)

[Chatbot 18](#_Toc110367172)

[Conversation algorithm 18](#_Toc110367173)

[Chatbot Configuration 22](#_Toc110367174)

[GUI 23](#_Toc110367175)

[System evaluation 23](#_Toc110367176)

[Testing 23](#_Toc110367177)

[Emotions classification testing 23](#_Toc110367178)

[Quotes responses testing 26](#_Toc110367179)

[Results 27](#_Toc110367180)

[Future Work 27](#_Toc110367181)

[Conclusion 27](#_Toc110367182)

[References 27](#_Toc110367183)

# Overview

In this report, a custom chatbot implementation is proposed that will respond to the user ‘s input either according to the detected emotion or with a quote that is close to the user’s input messages.

# Problem Formulation

Life is stressful and we all find ourselves sometimes in dark places, needing someone to talk to and share our feelings.  Sometimes we can’t find a person to talk to, but we always have our phones on our hands. for these reasons, a chatbot that will assess the current user’s input text emotion and respond accordingly to improve the users’ mood would definitely ease the users’ feelings. For example, If the user is feeling depressed the chatbot should suggest positive activities or send positive messages to make the user’s feel better. The proposed system is inspired by this paper[1]

# System Overview

The system overview is as presented in the following graph

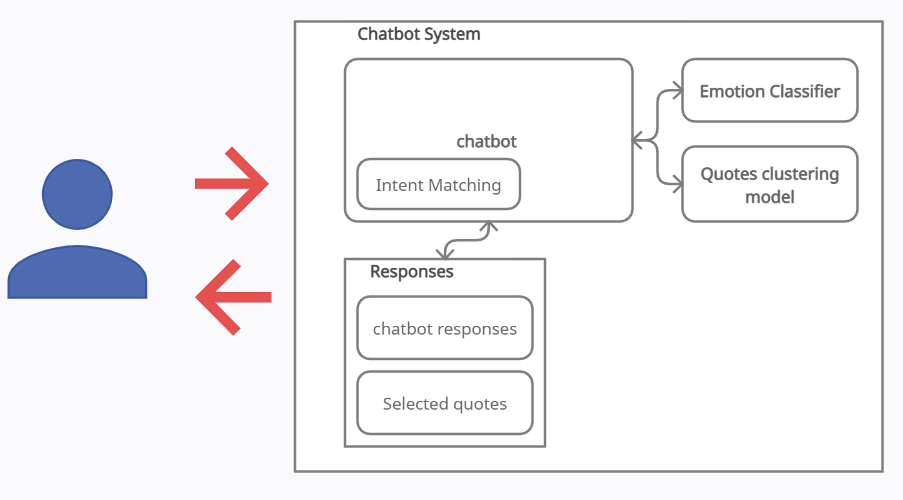


Figure - Proposed system overview

## Conversation Flow

To make the user feel like he is talking to a real person rather than a bot, the following conversation flow is proposed:

1. Greeting the user
2. Asking the user some friendly questions to get to know the user better, for example ask the user for his name and whether he likes to do specific activities.
3. Invoke the user to start talking.
4. Keep conversation going by asking about topics or objects that the user mentioned
5. Analyze user input messages and decide on appropriate response.
6. Respond to the user

## System Components

By analyzing the proposed conversation flow, the following components were identified:

1. **Emotion classifier:** a model that will be used to classify the users input and predict its emotion.
2. **Quotes clustering model:** a model that is used to cluster the selected set of quotes to be aid in retrieving a quote that is similar to the user’s input messages.
3. **Chatbot:** the agent that will collect the user’s input messages, analyze them and generate appropriate response to the user.

## Datasets

The following datasets were used to build the system:

### Emotions dataset

A dataset contains a collection of documents and their emotions and is available on Kaggle in this link.

### Quotes dataset

A dataset that contains a big collection of quotes with their authors, category, tags, and popularity. It is available on Kaggle in this link.

### Bot responses dataset

A small dataset that was collected by searching the internet manually for appropriate responses for every emotion type of the types specified in the emotions dataset.

# Components Implementation

## Emotions classifier

This model was created using emotions dataset, according to the following steps:

### Loading dataset

Loading dataset “training, validation, testing” files into dataframes.

Graphical user interface, text, application, email

Description automatically generated

Figure - Code snippet for loading emotions dataset files into dataframes

Then dropping love emotion Sentences from datasets since the basic human emotions include (joy, sadness, fear, anger, disgust, surprise), since love is not included it will be discarded.

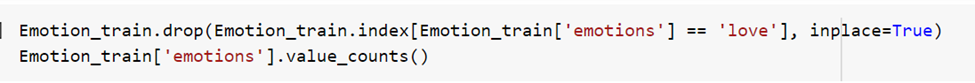


Figure - Code snippet for dropping records of "love" emotion

**Encoding emotions column**

Categorical variables need to be transformed into numbers to be used by the model, since most of them accept only numerical values as inputs. The mapping was done as follows:

Text

Description automatically generated

Figure - Code snippet for encoding emotions string labels

The column “label” represents the label for every emotion. The values after encoding are displayed in the following figure.

Text

Description automatically generated with low confidence

Figure - Preview of training dataframe after encoding emotions

### Data visualization

1. Distribution

**Distribution of the emotions in the training set to represent the balance of class,** balancing data gives us the same amount of information to help predict each class and therefore gives a better idea of how to respond to test data.

A picture containing text

Description automatically generated

Figure - Code snippet for creating a count plot of emotions classes

All the emotions have equal number of classes in train, validation and test. but the classes distribution itself is unbalanced.

Chart, bar chart, histogram

Description automatically generated

Figure - Emotions classes distribution for training, validation, and testing sets

1. Checking text length of every emotion.

Text

Description automatically generated

Figure - Code snippet for displaying text length for every emotions classes for training, validation and testing sets

Nearly all the text length has the same value for all the classes and across different dataset partitions.

Chart, line chart, histogram

Description automatically generated

Figure - Text length of emotions classes for training, validation, and testing sets

1. Wordcloud

Word cloud is used for representing for each emotion in which the size of each word indicates it.

The following plots display a word cloud of 75 every emotion’s keywords.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |
|  | |

Some words like (feeling, feel) are repeated in almost all emotions word clouds, but these words won't help in classification of emotions.

1. N-Grams

Applying n-grams to understand which words mainly occur together. For instance, we look at the distribution of unigrams, bigrams, and trigrams across emotions.

Here is the code to calculate n-grams and **calculating unigrmas, bigrams and trigrams for all given emotions.**

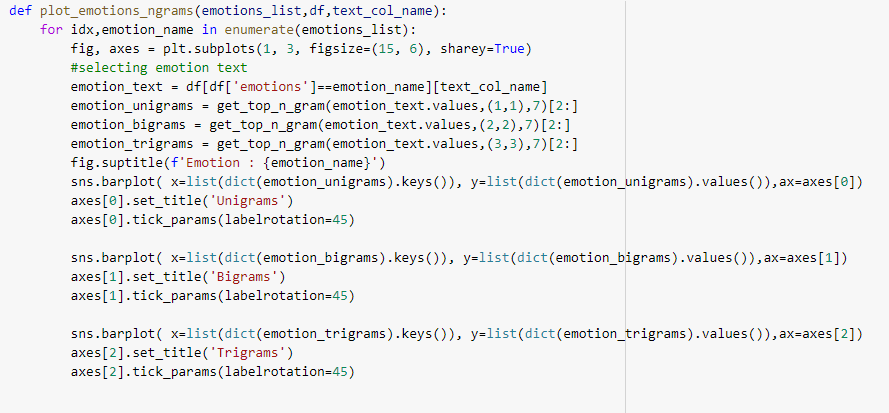


Figure - Code snippet for generating n-grams plots

**N-Gram for training set before data cleaning.**

|  |
| --- |
|  |
| Figure - "sadness" emotion n-grams from the training set |

The graph shows that there are many repeated words -such as “im” and “just”- that are not important and do not express a specific label, so data preparation to drop and remove these words is crucial. Also, the bigrams seem to identify the emotion classes better.

The following are the n-grams plots after cleaning the data.

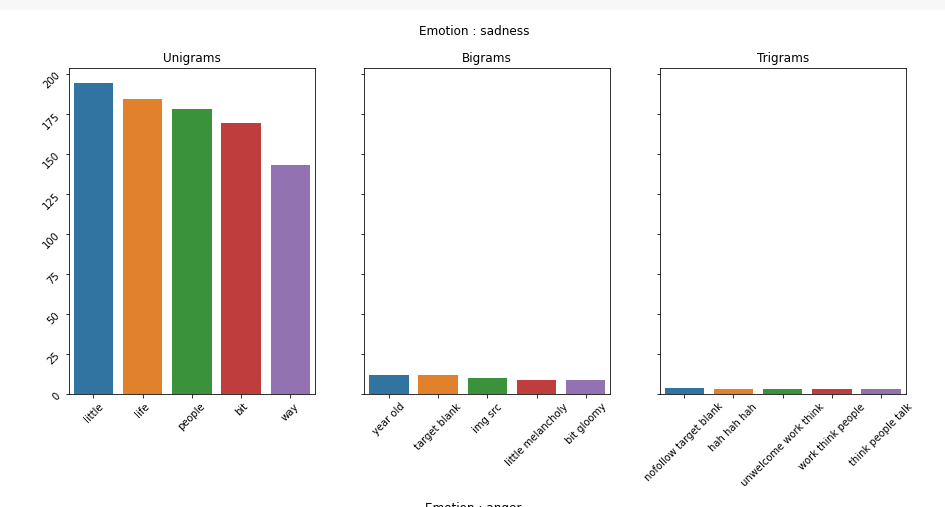


Figure – “sadness" emotion n-grams of cleaned text from the training set

### Data preparation

As shown in N-Gram graph before preprocessing, the dataset contains unnecessary words which will definitely degrade the model’s performance, so data cleaning was performed to improve data quality. The cleaning process includes the following action:

1. Lemmatization: to return words into their root form.
2. Removing punctuations: to minimize ambiguity.
3. Removing stop words: to minimize noise.
4. Removing unwanted words to reduce noise.

The final cleaned version of the training dataset text after cleaning can be seen in the column “cleantext”.

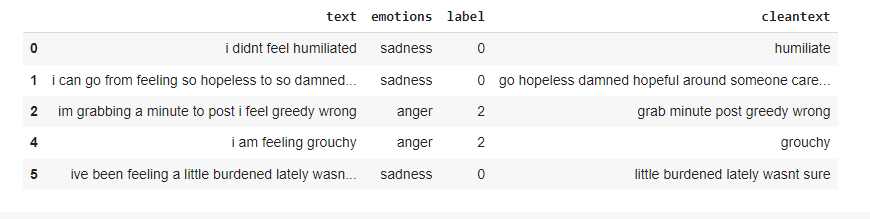


Figure - Preview of the cleaned text of the training set

### Classification models

The dataset was already split into 3 files, one for training, one for validation and one for testing. So, the training set was loaded and used in training the classifiers and the validation set was used to compare classifiers performance and choose the best model while the test set was used to get champion model accuracy.

Feature engineering was performed using TF-IDF with min\_df = 15 to minimize selected word features and ngram\_range = (1,2) to include both unigrams and bigrams in selected features- because from plotting N-grams bi grams appeared to may help in classifying emotions-.

Three pipelines using three different classification models “SVM, NB, LR” were created and trained using 10 folds cross validation. The following are the graphs show the cross-validation accuracies

|  |  |
| --- | --- |
|  |  |
| Average Accuracy: 0.85 (+/- 0.01) | Average Accuracy: 0.82 (+/- 0.02) |
|  | |
| Average Accuracy: 0.85 (+/- 0.01) | |

Here are two models that have equal 10-fold average accuracy, so validation dataset was used to find and the best model between SVM & LR through their classification report.

**SVM classification report:**

Table

Description automatically generated

Figure - SVM classification report on validation set

**LR classification report:**

Table

Description automatically generated

Figure - LR classification report on validation set

Precision and recall of SVM are higher than LR which means that SVM maximized the numbers of true positives. Also, SVM has a higher f1-score which is better because the data was unbalanced so comparing the overall accuracy is not enough.

### Champion model analysis

the champion model is SVM, and by evaluating the performance of it using test dataset, the final accuracy = 87. The following is the confusion matrix and classification report of testing the model using the test set.

A picture containing graphical user interface

Description automatically generated

Figure - SVM classification report on testing set

The champion model pipeline was used later for emotion prediction.

### Error analysis

**Wrong predictions**

The following dataframe displays the wrong predictions and their incorrectly predicted label

Graphical user interface, text, application, email

Description automatically generated

Figure - Preview of wrong predictions from testing set

It appears that there are 233 records that were falsely classified.

**Class count of wrong predictions**

Chart, bar chart

Description automatically generated

Figure - falsely predicted emotions from testing set class count

From the wrong predictions correct label class count, it is seen that most of the wrong predictions belong to “anger” and “sadness” emotions.

**Word cloud of wrong predictions**

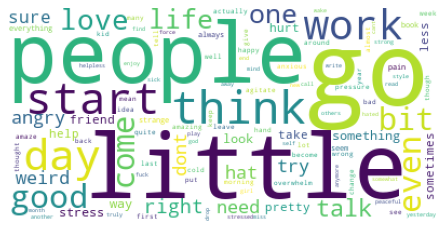


Figure - word cloud for misclassified records of testing set

From the word cloud, the most repeated words are unbiased to a specific emotion. Except for the word “angry” which should clearly indicate “anger” emotion.

**N-grams of wrong predictions of “anger”**

**Chart, bar chart, waterfall chart

Description automatically generated**

Figure - N-grams of misclassified "anger" emotion text from testing set

**N-grams of wrong predictions of “sadness”**

**Chart, bar chart

Description automatically generated**

Figure - N-grams of misclassified "sadness" emotion text from testing set

From looking at both n-grams, it can be noticed that most of the unigrams are unbiased and can classified into different emotions, but for the bigrams they show clearly show the emotion of the text, yet their frequency is very low, so these terms were probably ignored because of the configured TF-IDF parameters where min\_df = 15.

## Quotes clustering model

This model was created using quotes dataset, according to the following steps:

### Loading dataset

The dataset consists of a single json file that was loaded and used in model training. The following figure shows a dataframe containing the loaded dataset. The dataset contained 48391 quotes.

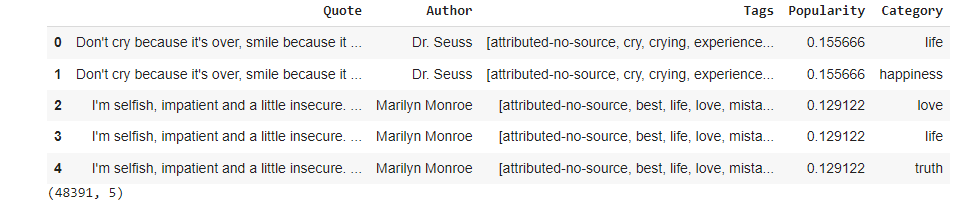


Figure - Preview of quotes dataset

The dataset was checked to see if it contained any na values in columns or rows. Also, the quotes categories were checked to see if there were quotes with an empty category.

Text

Description automatically generated

Figure - Code snippet for finding NA and uncategorized quotes counts

The uncategorized quotes were dropped

### Data visualization

**Quotes count per category plot**

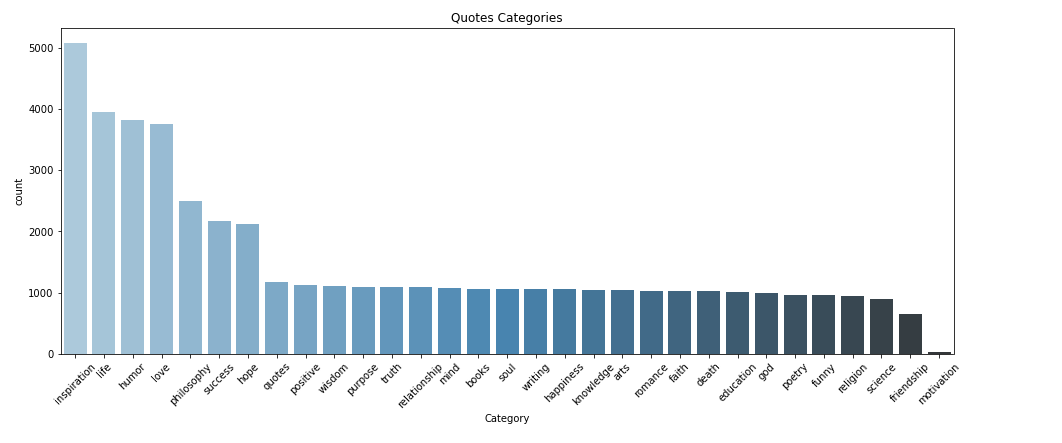


Figure - Number of quotes per category

On checking some of the quotes, it was noticed that some of them were of unknown authors while others were taken from books without considering their meaning out of the book’s context. So, the dataset was filtered to reduce the number of quotes and get quotes with high quality only.

After considering the authors, quotes popularity and quotes count per author. Only the quotes that belonged to the authors were kept.

Text

Description automatically generated

Figure - List of selected Authors

The selected quotes count was 1694.

**Selected quotes count per category plot**

**Chart, waterfall chart

Description automatically generated**

Figure - Number of quotes per category for the selected authors

### Data preparation

The same cleaning process used in preparing the emotions dataset text was applied on the quotes text, the cleaning steps were as follows:

1. Lemmatization: to return words into their root form.
2. Removing punctuations: to minimize ambiguity.
3. Removing stop words: to minimize noise.
4. Removing unwanted words to reduce noise.

And the final cleaned quote text can be seen in “cleanQuote” column in the following dataframe figure.

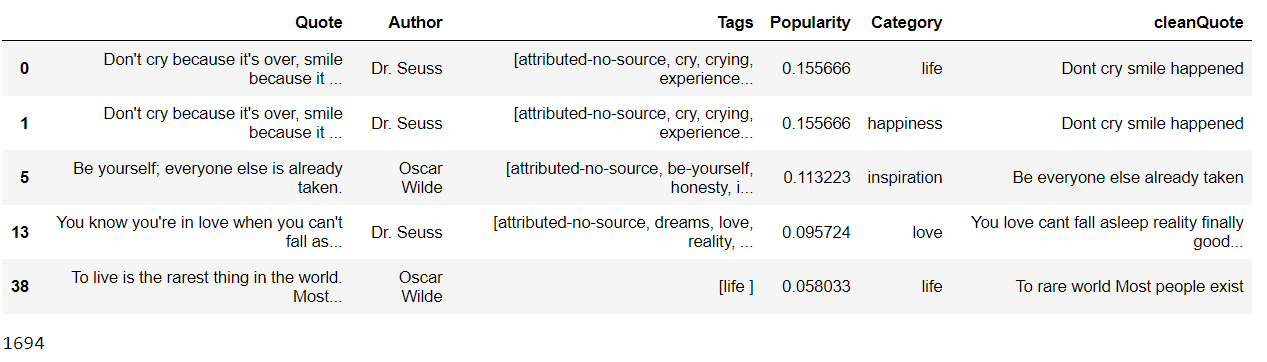


Figure - Preview of the quotes after cleaning its text

### Feature Engineering

The quotes emotion was predicted and saved to the dataframe using the emotions classifier, this feature will later be used by the chatbot to help in narrowing down the nominated quotes to select a response to the user from.

### Modelling using K-Means

Feature engineering was performed using TF-IDF with min\_df = 30 to minimize selected word features and ngram\_range = (1,2) to include both unigrams and bigrams in selected features for obtaining better homogenous clusters.

#### Select best number of clusters

Using the previously extracted features, the following graphs of WCSS and Silhouette were created to obtain the best number of clusters.

A picture containing chart

Description automatically generated

Figure - Plots for selecting the best number of clusters for K-means

The best number of clusters is 17.

A pipeline was created using mentioned TF-IDF configuration and K-Means clustering model as follows:

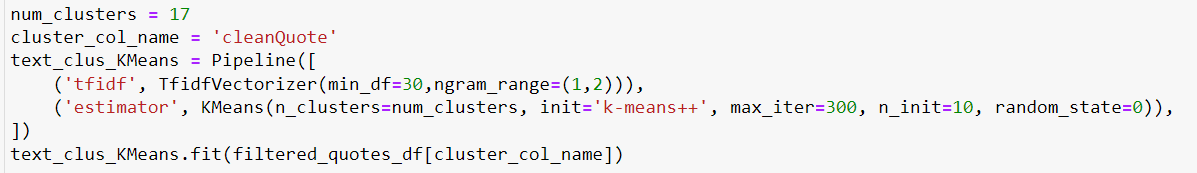


Figure - Code snippet for clustering model pipeline

The pipeline allowed for easier integration when using the clustering model.

#### Clusters evaluation

The resulting clusters were plotted using t-sne

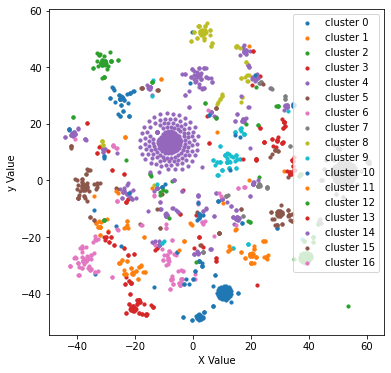


Figure - t-sne plot of K-means clusters

It can be seen that some clusters points are close to each other while other points are scattered. But the overall quality of the clusters scatter seems acceptable.

## Chatbot

To get the ability of changing the conversation flow and allow the chatbot to probe the user about the entities mentioned in his words throughout the conversation, a custom implementation of the chatbot was developed. The following describes the different components of this agent.

### Conversation algorithm

The conversation algorithm is divided into two major parts: intent matching and conversation flow.

#### Intent matching

Following the suggested method described [here](https://online.datasciencedojo.com/blogs/rule-based-chatbot-in-python/) for intent matching, an intent matching algorithm was developed using regular expressions. For every intent a set of keywords are provided and a set of sentences that should be used to identify this intent. Then when the chatbot system is starting a list of synonyms is created for the keywords using nltk library. Regular expressions are constructed such that if the intent’s sentences or keywords or synonyms were identified in the user text, the corresponding intent is selected. This is an example of the regular expression generated for “goodbye” intent.

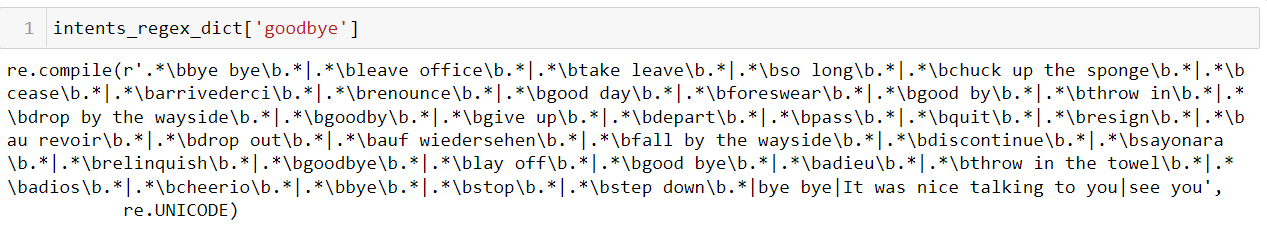


Figure - regex patterns for "goodbye" intent

#### Conversation flow

##### Chatbot agent state and user intents

In addition to the user intent identified in user input messages, the chatbot contains a state as well. this state helps the chatbot in deciding if it should invoke the user to talk or reply with a static response from the provided list of responses for every intent. In the current implementation the chatbot decides what to do based on the detected user intent which is considered the chatbot state.

##### Basic conversation flow

The following flowchart describes the sequence followed by the chat bot to talk and respond to the user

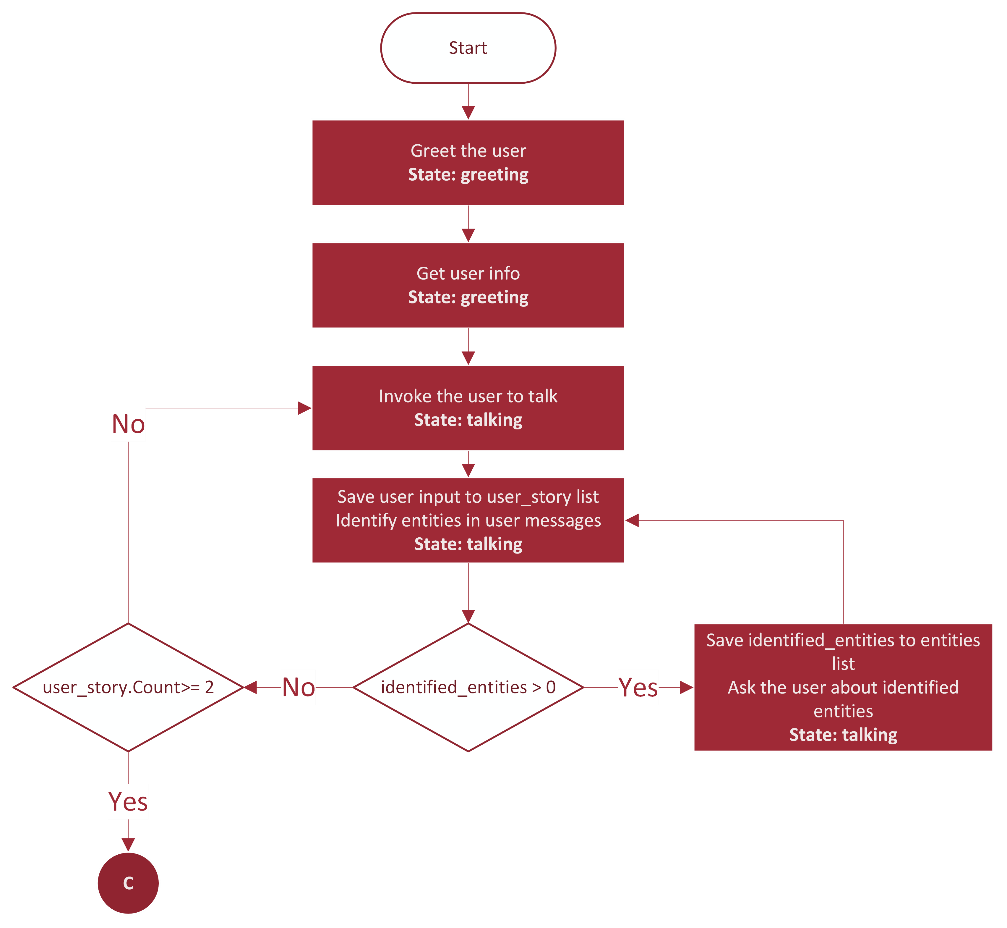


Figure - Flowchart for the "talking" state of the chatbot agent

The “user\_story” list will contain the group of input user messages that the chatbot will decide on the response from, these sentences will be cleaned using the same technique used in cleaning the text before being passed to emotion classifier or quotes clustering models. The reason for collecting multiple sentences is to increase the accuracy of predicted emotion since most of the emotions text length of the emotions dataset was in the range between 50 to 100, which is usually not going to be the length of a single input message from the user.

Entities from user input messages are recognized using Spacy NER model for English language (en\_core\_web\_sm), the recognized entities are added to “identified\_entites” list and the user is invoked to talk about them one by one. While the user is talking newly recognized entities are added to the list – if the entity already exists in the list, it won’t be added again-, and each time the chatbot asks the user about a specific entity, this entity is removed from the list.

##### Response selection

The response selection process happens as follows:

Diagram

Description automatically generated

Figure - Flowchart for response selection process of the chatbot agent

The users answer to liking reading questions is used to increase the probability of responding to the user using a quote, because if the user likes reading, he will probably like being given a quote response. Also, in case the chatbot decides to respond with a quote yet the selected quote cluster does not have any quotes that match the users input emotion, any random quote from the cluster should be returned to the user.

When a response is selected and displayed to the user, the “user\_story” list will be cleared.

At the end the user may choose to continue talking, in this case the state is preserved as “talking” and the chatbot invokes the user to talk again.

##### State update

While the user is talking to the chatbot and the active state is “talking”, the user input is constantly monitored against intents and if a new intent match is found the state is switched to the appropriate intent and state, and an appropriate response is selected from the matched intents responses.

##### Debugging the conversation flow

The conversation flow can be monitored by looking at the notebook cell output. The state, identified entities, user input list and identified emotion are all printed. The following is an example of a conversation and the snippet of the corresponding printed log messages.

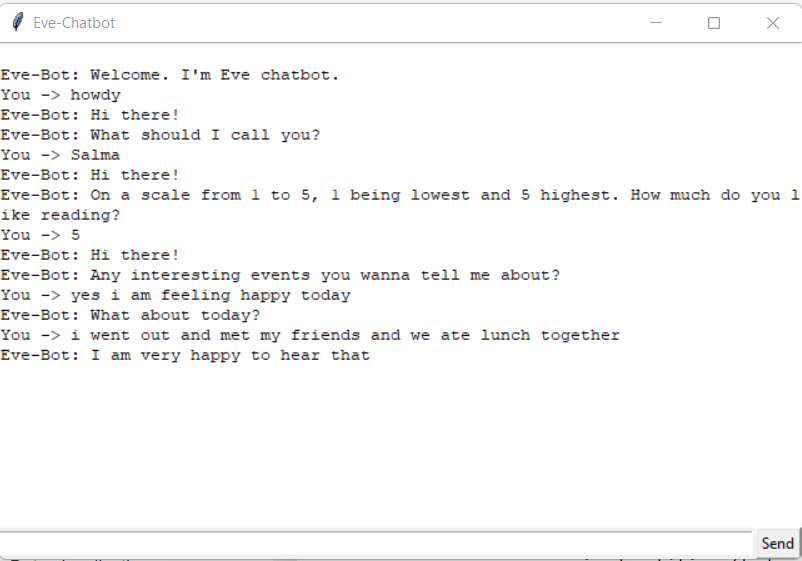


Figure - chatbot conversation example for "joy" emotion

Text

Description automatically generated

Figure - chatbot printed log for the "joy" emotion example

### Chatbot Configuration

To allow flexible development the chatbot intents and intent responses are provided as key value pairs that can easily be read from json files or added statically in code. The following are the required configuration data description:

* **Intents dictionary:** a dictionary that has intent name as its key, and the value is a list containing the following items:
  + **key\_words:** list of key words that their synonyms are generated, and both the keywords and synonyms are used identify this intent using regex matching.
  + **sentences:** list of sentences that are used in regex patterns matching to detect the intent of the user input.
  + **responses:** list of possible responses for the intent
  + **patterns:** list of regex patterns constructed by the chatbot based on intent’s key words and their synonyms to be used in detecting the intent of the user input.

Text

Description automatically generated

Figure - Current intent dictionary for the chatbot agent

* **Bot intent responses**: a dictionary that has a few static responses used by the chatbot during conversation in the ‘talking’ state, these messages don’t include emotion-based responses. The emotion-based responses are selected from “chatbot-responses.csv”.

Text

Description automatically generated with medium confidence

Figure - Current static responses for the chatbot agent

### GUI

A simple GUI was created using python tkinter library, based on this [link](https://www.analyticsvidhya.com/blog/2021/07/build-a-simple-chatbot-using-python-and-nltk/). The GUI was updated to allow the user input to be sent to the chatbot system when pressing enter after typing the user’s message.

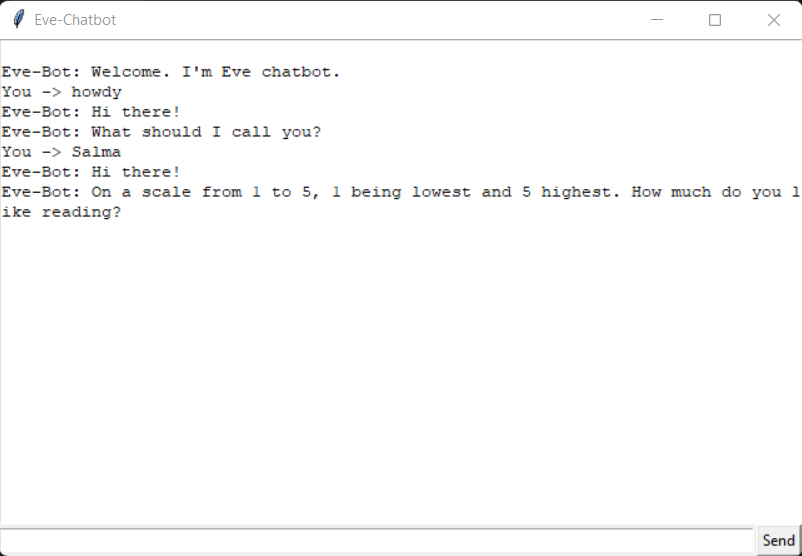


Figure - chatbot system GUI

# System evaluation

## Testing

To facilitate the testing, if the user answers to liking reading with 0, then selected response will be from “chatbot\_responses” dataset.

### Emotions classification testing

The following images were acquired while testing the chatbot response for different emotion.

#### Joy

Graphical user interface, text, application, email

Description automatically generated

Figure - Chatbot conversation example for "joy" emotion

#### Sadness

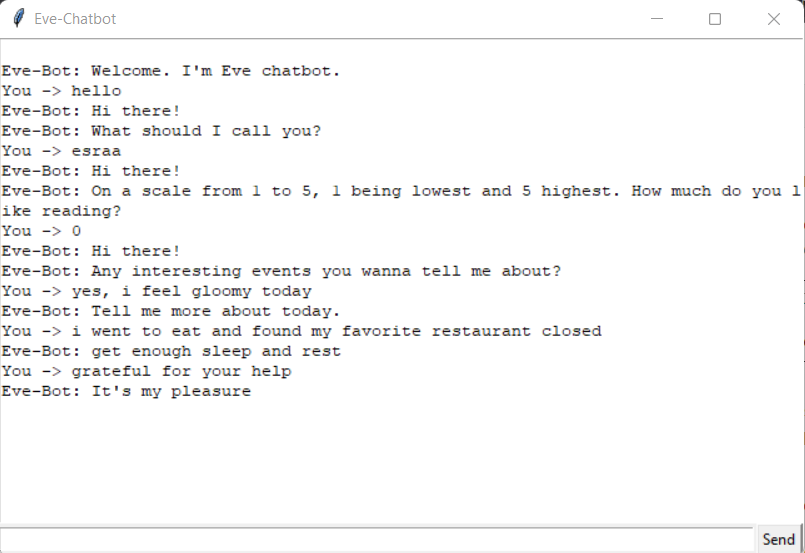


Figure - Chatbot conversation example for "sadness" emotion

#### Fear

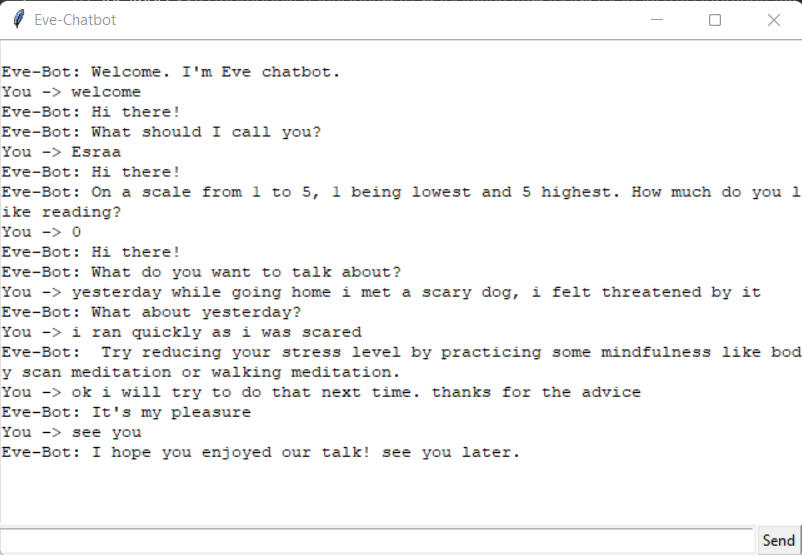


Figure - Chatbot conversation example for "fear" emotion

#### Surprise

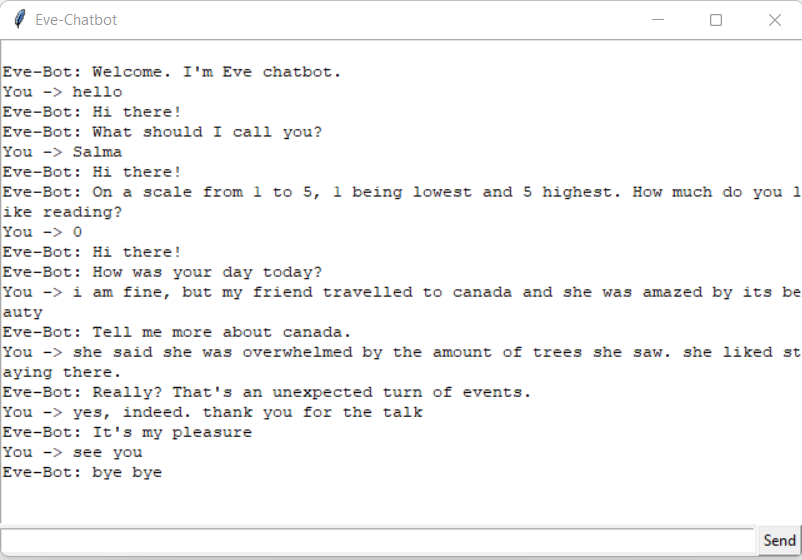


Figure - Chatbot conversation example for "surprise" emotion

#### Anger

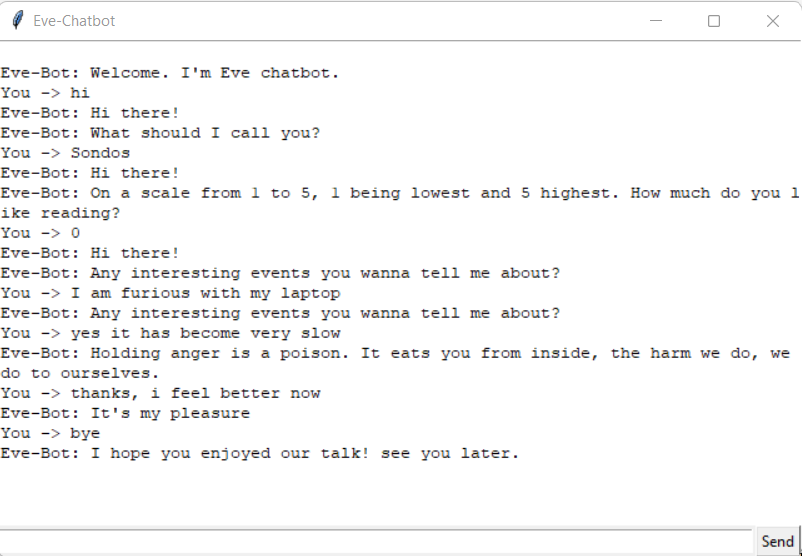


Figure - Chatbot conversation example for "anger" emotion

### Quotes responses testing

The following images were acquired while testing the chatbot similar quote response.

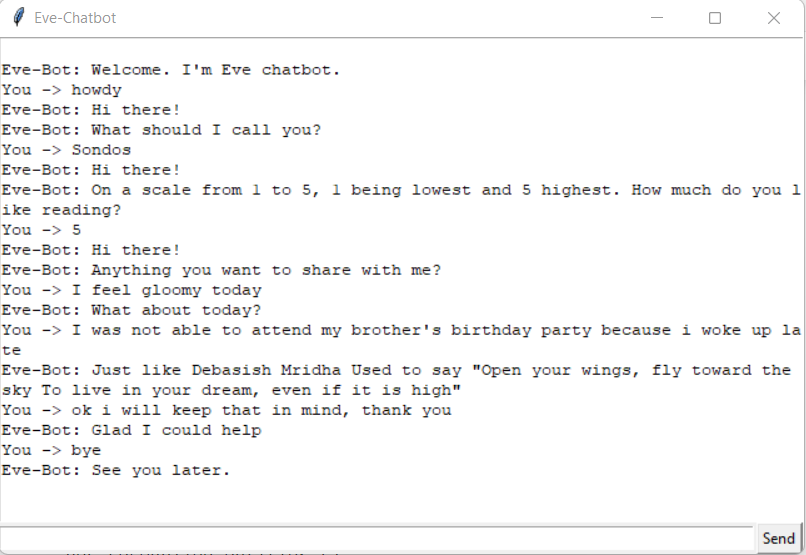


Figure - Chatbot conversation example for quote response

## Results

From the previous conversation, it can be seen that the chatbot correctly identified the emotion, yet some responses felt a little bit out of context. It is also noticed that the chatbot keeps greeting the user multiple time before getting into talking state. For the quote integration the responses were sometimes not very relevant to the user input messages.

# Future Work

User input processing:

* Auto correct user input messages before analyzing them by the chatbot system.
* Validate the user’s input, for example when asking for a numeric value, text input should be rejected.

Chatbot responses:

* Expand the current dataset by adding more responses
* Consider creating templating responses where the recognized user entities can be inserted in specific positions in the response text.
* Save the users answers to welcoming questions to avoid making him answer the same questions every time the chatbot is launched.

Chatbot algorithm:

* Consider maintaining both the chatbot state and the user intent and use both of them to select more appropriate chatbot responses.
* Recognize the users input messages context in an attempt of detecting the end of context, to help in determining when the chatbot should select a response based on the emotion or probe the user for more information.
* Improve intent matching by using NLG models to expand the sentences provided for the intent and then using a machine learning model to get the corresponding intent to the user text.

# Conclusion

A custom chatbot implementation is proposed, it incorporates several machine learning models to try and simulate a humane conversation that will help the user have a better emotional state.

# References

[1] S. Moulya and T. R. Pragathi, “Mental Health Assist and Diagnosis Conversational Interface using Logistic Regression Model for Emotion and Sentiment Analysis,” *Journal of Physics: Conference Series*, vol. 2161, no. 1, p. 012039, Jan. 2022, doi: 10.1088/1742-6596/2161/1/012039.