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

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

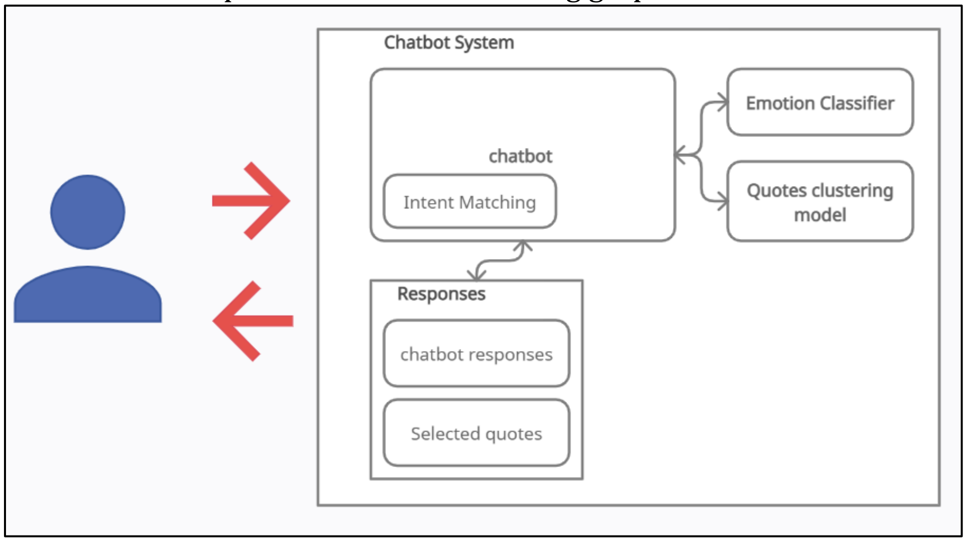


Figure 1- Proposed system overview

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.

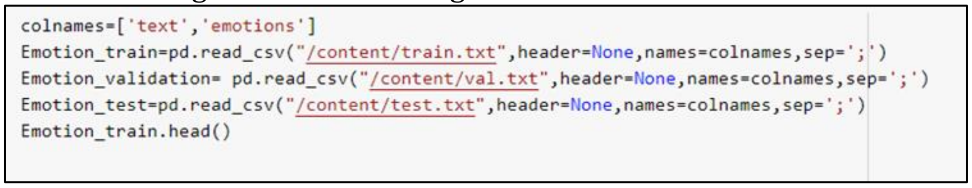
Components Implementation

Emotions classifier

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

Loading dataset

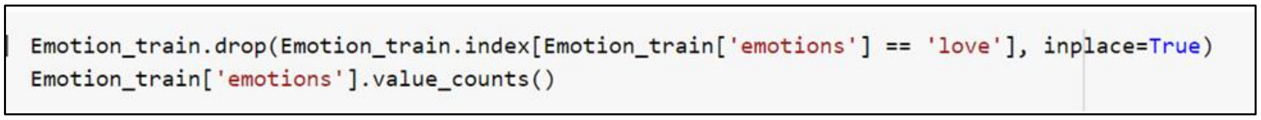
Loading dataset "training, validation, testing" files into dataframes.



```
colnames=['text','emotions']
Emotion_train=pd.read_csv("/content/train.txt",header=None,names=colnames,sep=';')
Emotion_validation= pd.read_csv("/content/val.txt",header=None,names=colnames,sep=';')
Emotion_test=pd.read_csv("/content/test.txt",header=None,names=colnames,sep=';')
Emotion_train.head()
```

Figure 2 - Code snippet for loading emotions dataset files into dataframes

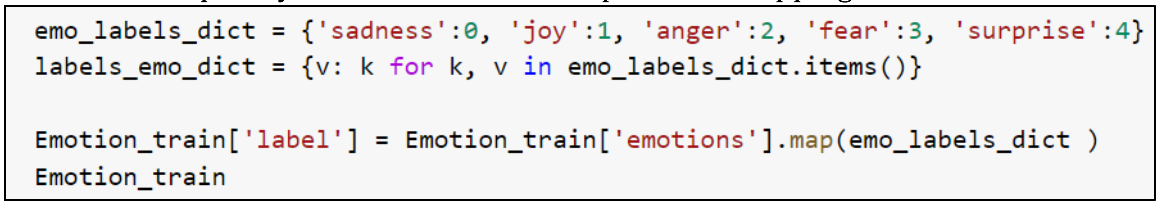
Then dropping love emotion Sentences from datasets since the basic human emotions include (joy, sadness, fear, anger, disgust, surprise), since love is not included it will be discarded.



```
Emotion_train.drop(Emotion_train.index[Emotion_train['emotions'] == 'love'], inplace=True)
Emotion_train['emotions'].value_counts()
```

Figure 3 - Code snippet for dropping records of "love" emotion

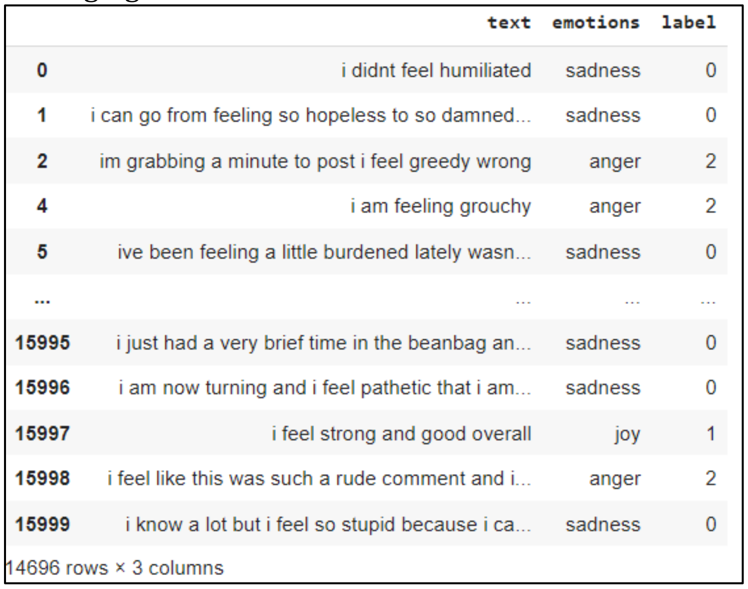
Encoding emotions column
Categorical variables need to be transformed into numbers to be used by the model, since most of them accept only numerical values as inputs. The mapping was done as follows:



```
emo_labels_dict = {'sadness':0, 'joy':1, 'anger':2, 'fear':3, 'surprise':4}
labels_emo_dict = {v: k for k, v in emo_labels_dict.items()}

Emotion_train['label'] = Emotion_train['emotions'].map(emo_labels_dict )
Emotion_train
```

Figure 4 - 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	emotions	label
0	i didnt feel humiliated	sadness	0
1	i can go from feeling so hopeless to so damned...	sadness	0
2	im grabbing a minute to post i feel greedy wrong	anger	2
4	i am feeling grouchy	anger	2
5	ive been feeling a little burdened lately wasn...	sadness	0
...	...	...	...
15995	i just had a very brief time in the beanbag an...	sadness	0
15996	i am now turning and i feel pathetic that i am...	sadness	0
15997	i feel strong and good overall	joy	1
15998	i feel like this was such a rude comment and i...	anger	2
15999	i know a lot but i feel so stupid because i ca...	sadness	0

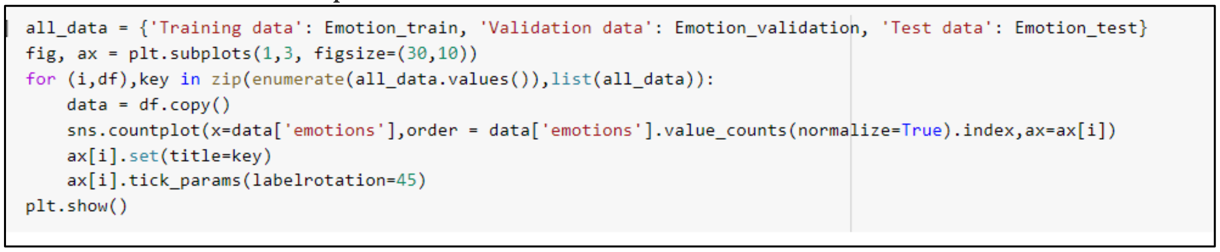
14696 rows × 3 columns

Figure 5 - 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.



```
all_data = {'Training data': Emotion_train, 'Validation data': Emotion_validation, 'Test data': Emotion_test}
fig, ax = plt.subplots(1,3, figsize=(30,10))
for (i,df),key in zip(enumerate(all_data.values()),list(all_data)):
    data = df.copy()
    sns.countplot(x=data['emotions'],order = data['emotions'].value_counts(normalize=True).index,ax=ax[i])
    ax[i].set(title=key)
    ax[i].tick_params(labelrotation=45)
plt.show()
```

Figure 6 - 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.

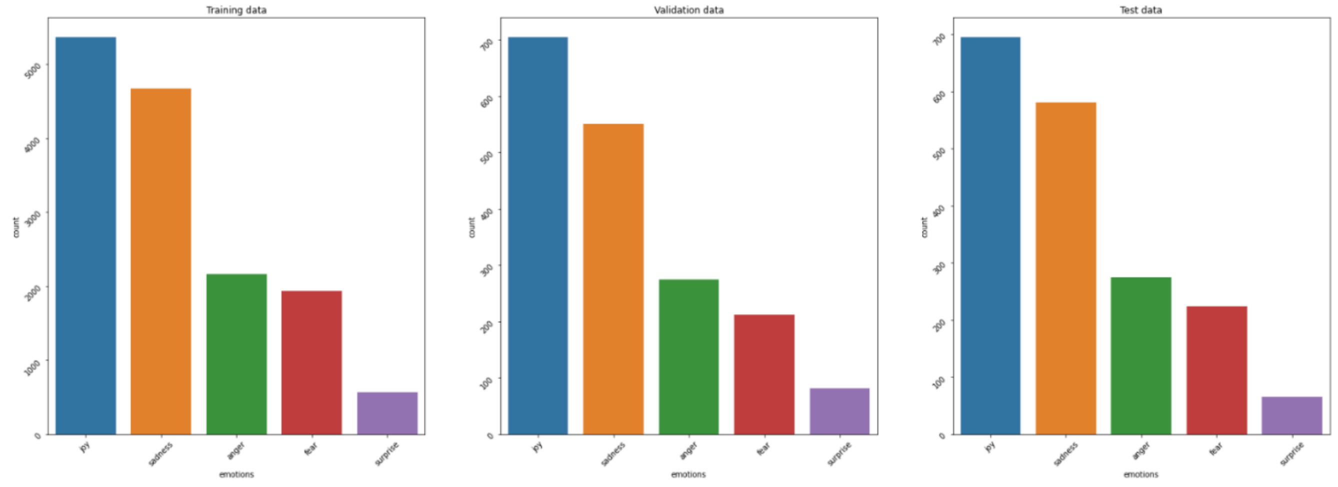
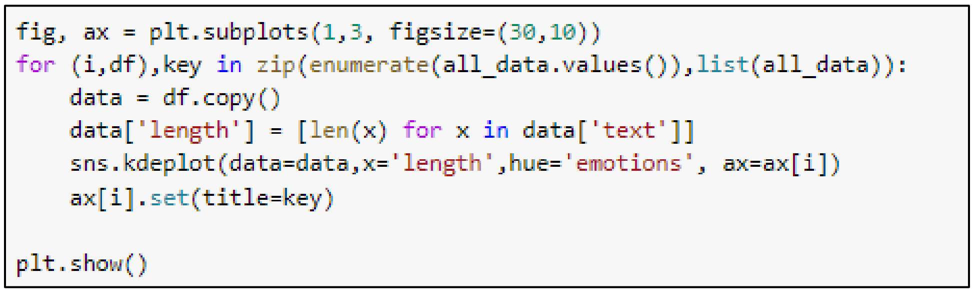


Figure 7 - Emotions classes distribution for training, validation, and testing sets

2- Checking text length of every emotion.



```
fig, ax = plt.subplots(1,3, figsize=(30,10))
for (i,df),key in zip(enumerate(all_data.values()),list(all_data)):
    data = df.copy()
    data['length'] = [len(x) for x in data['text']]
    sns.kdeplot(data=data,x='length',hue='emotions', ax=ax[i])
    ax[i].set(title=key)

plt.show()
```

Figure 8 - 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.

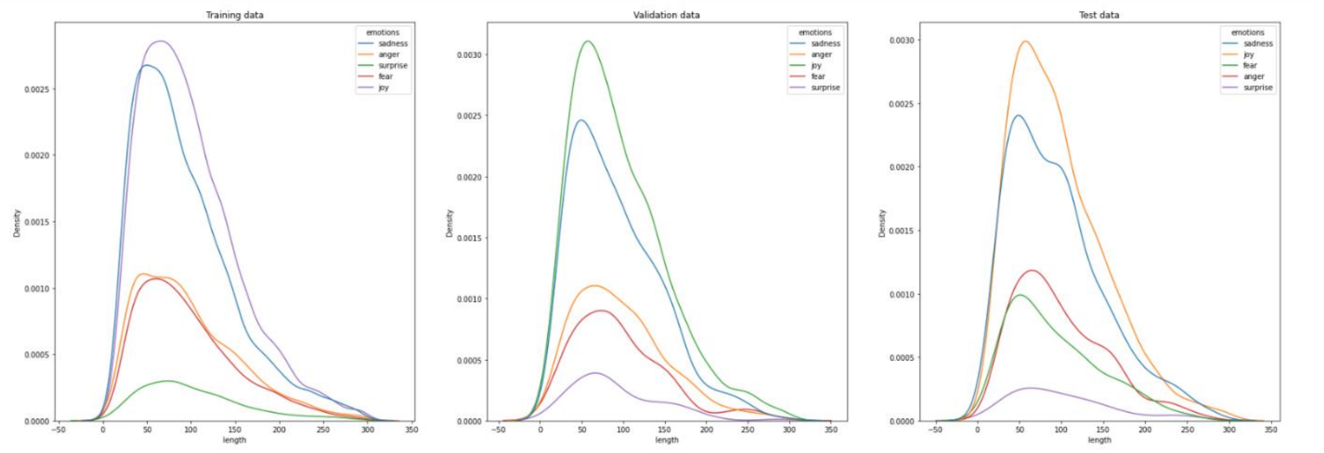
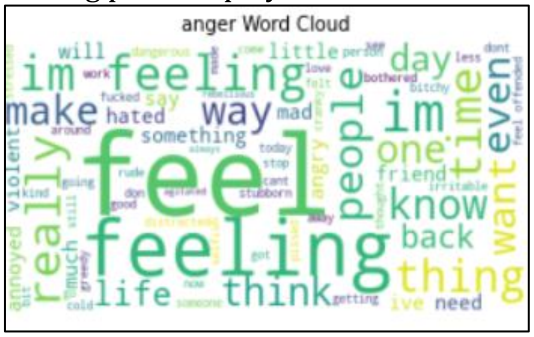
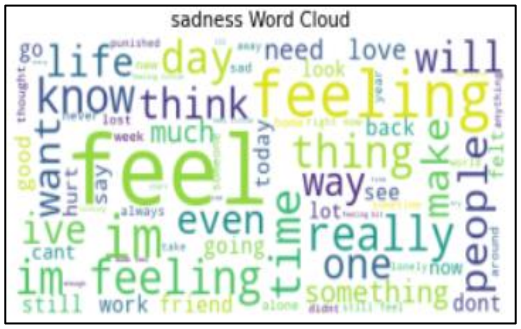


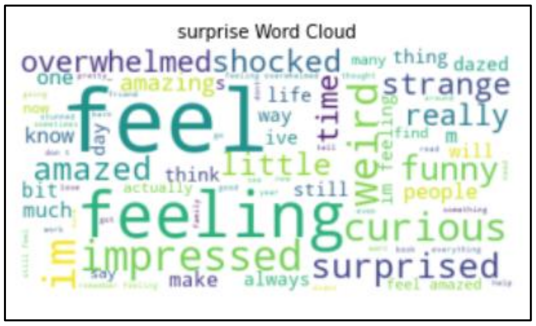
Figure 9 - Text length of emotions classes for training, validation, and testing sets

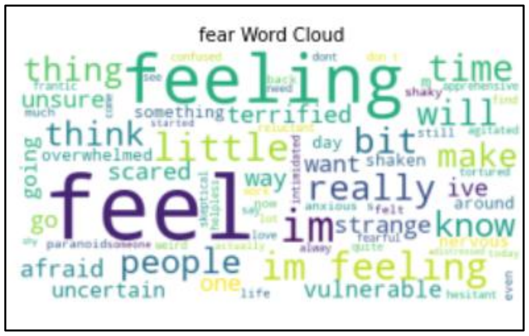
3- Wordcloud

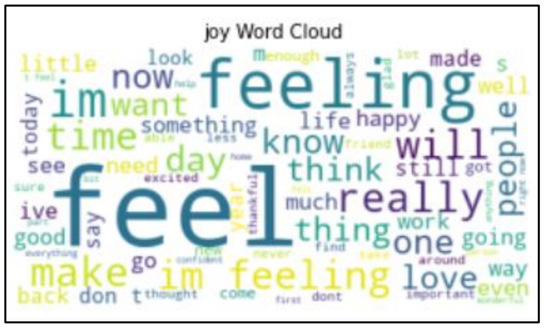
Word cloud is used for representing for each emotion in which the size of each word indicates it.
The following plots display a word cloud of 75 every emotion's keywords.









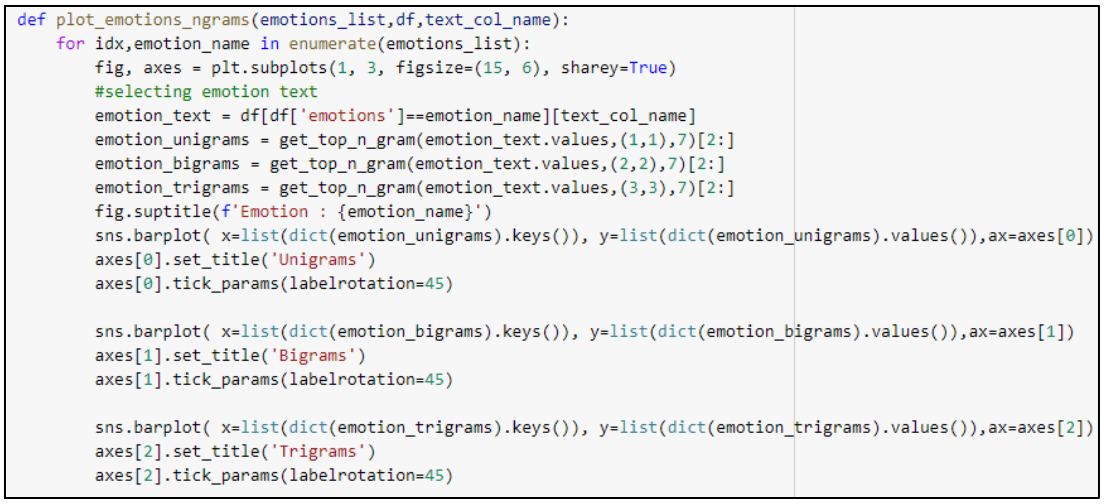


Some words like (feeling, feel) are repeated in almost all emotions word clouds, but these words won't help in classification of emotions.

4- 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.

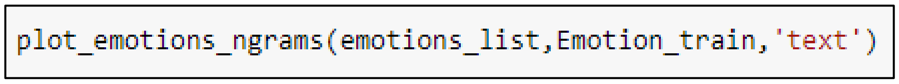
Here is the code to calculate n-grams and calculating unigrmas, bigrams and trigrams for all given emotions.



```
def plot_emotions_ngrams(emotions_list,df,text_col_name):
    for idx,emotion_name in enumerate(emotions_list):
        fig, axes = plt.subplots(1, 3, figsize=(15, 6), sharey=True)
        #selecting emotion text
        emotion_text = df[df['emotions']==emotion_name][text_col_name]
        emotion_unigrams = get_top_n_gram(emotion_text.values,(1,1),7)[2:]
        emotion_bigrams = get_top_n_gram(emotion_text.values,(2,2),7)[2:]
        emotion_trigrams = get_top_n_gram(emotion_text.values,(3,3),7)[2:]
        fig.suptitle(f'Emotion : {emotion_name}')
        sns.barplot( x=list(dict(emotion_unigrams).keys()), y=list(dict(emotion_unigrams).values()),ax=axes[0])
        axes[0].set_title('Unigrams')
        axes[0].tick_params(labelrotation=45)

        sns.barplot( x=list(dict(emotion_bigrams).keys()), y=list(dict(emotion_bigrams).values()),ax=axes[1])
        axes[1].set_title('Bigrams')
        axes[1].tick_params(labelrotation=45)

        sns.barplot( x=list(dict(emotion_trigrams).keys()), y=list(dict(emotion_trigrams).values()),ax=axes[2])
        axes[2].set_title('Trigrams')
        axes[2].tick_params(labelrotation=45)
```

Figure 10 - Code snippet for generating n-grams plots

N-Gram for training set before data cleaning.



```
plot_emotions_ngrams(emotions_list,Emotion_train,'text')
```

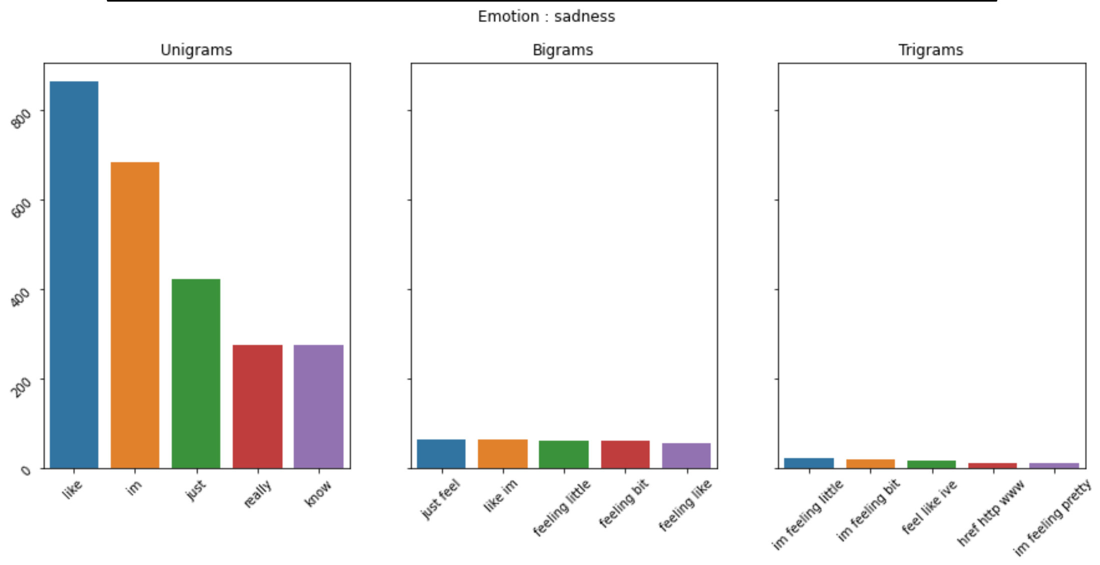


Figure 11 - "sadness" emotion n-grams from the training set

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.

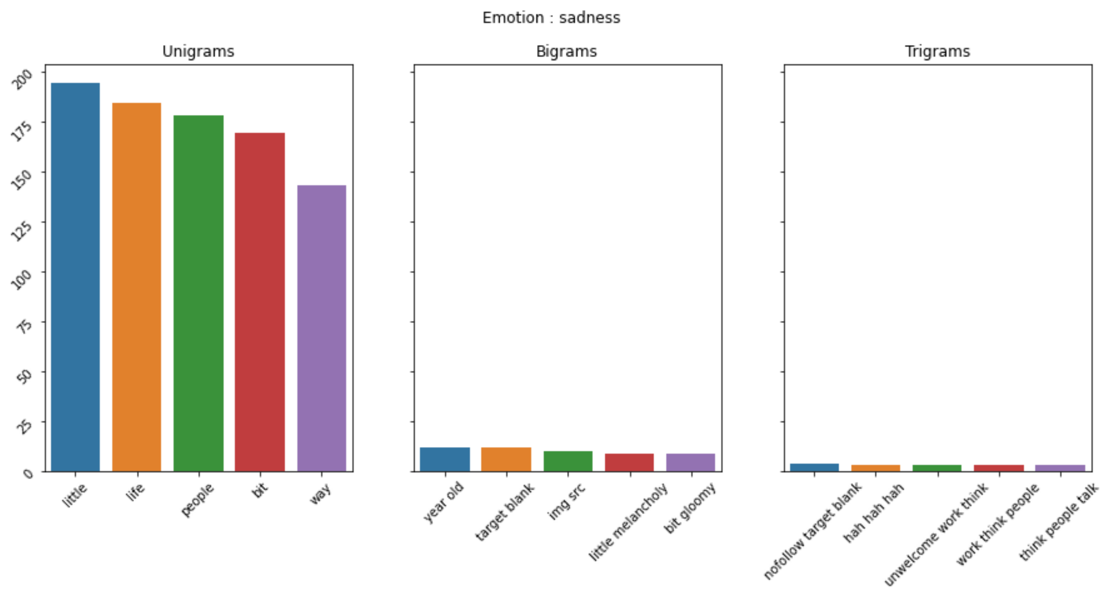


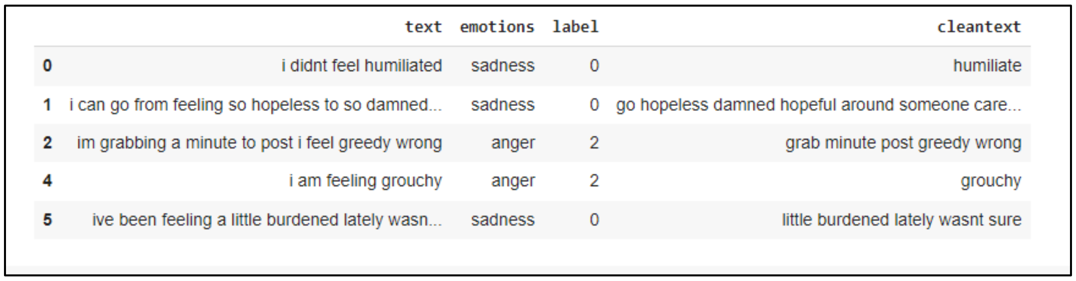
Figure 12 – "sadness" emotion n-grams of cleaned text from the training set

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".



	text	emotions	label	cleantext
0	i didnt feel humiliated	sadness	0	humiliate
1	i can go from feeling so hopeless to so damned...	sadness	0	go hopeless damned hopeful around someone care...
2	im grabbing a minute to post i feel greedy wrong	anger	2	grab minute post greedy wrong
4	i am feeling grouchy	anger	2	grouchy
5	ive been feeling a little burdened lately wasn...	sadness	0	little burdened lately wasnt sure

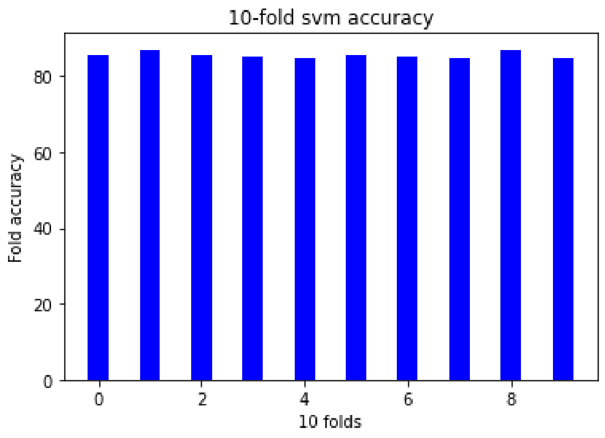
Figure 13 - Preview of the cleaned text of the training set

Classification models

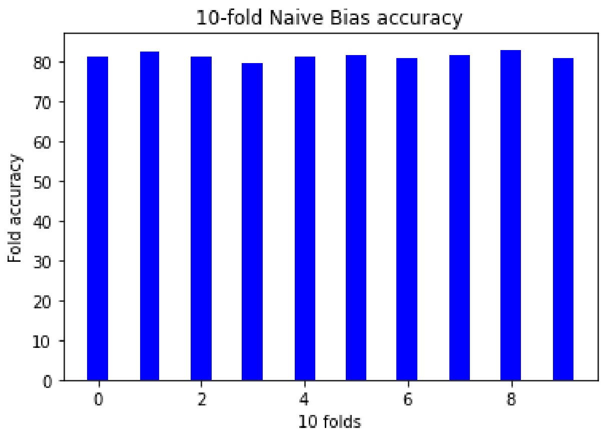
The dataset was already split into 3 files, one for training, one for validation and one for testing. So, the training set was loaded and used in training the classifiers and the validation set was used to compare classifiers performance and choose the best model while the test set was used to get champion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

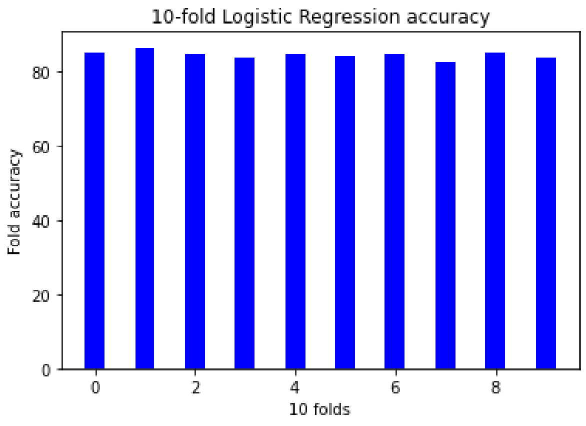
Three pipelines using three different classification models "SVM, NB, LR" were created and trained using 10 folds cross validation. The following are the graphs show the cross-validation accuracies



Average Accuracy: 0.85 (+/- 0.01)



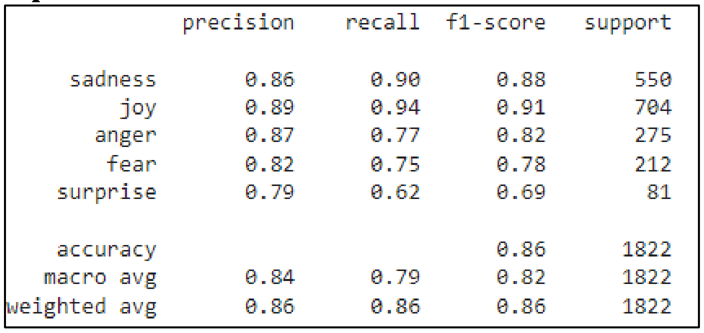
Average Accuracy: 0.82 (+/- 0.02)



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.

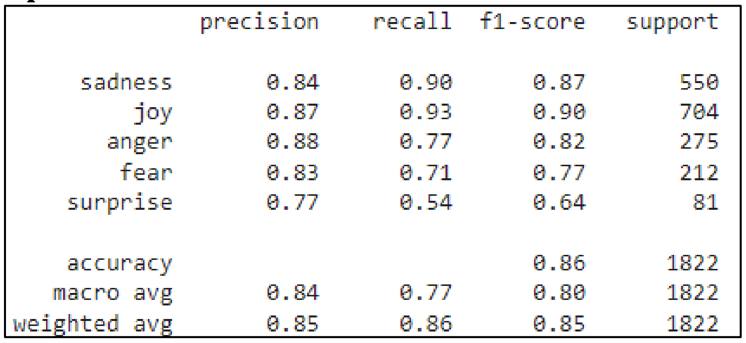
SVM classification report:



```
              precision    recall  f1-score   support

     sadness       0.86      0.90      0.88       550
         joy       0.89      0.94      0.91       704
       anger       0.87      0.77      0.82       275
        fear       0.82      0.75      0.78       212
    surprise       0.79      0.62      0.69        81

    accuracy                           0.86      1822
   macro avg       0.84      0.79      0.82      1822
weighted avg       0.86      0.86      0.86      1822
```

Figure 14 - SVM classification report on validation set

LR classification report:



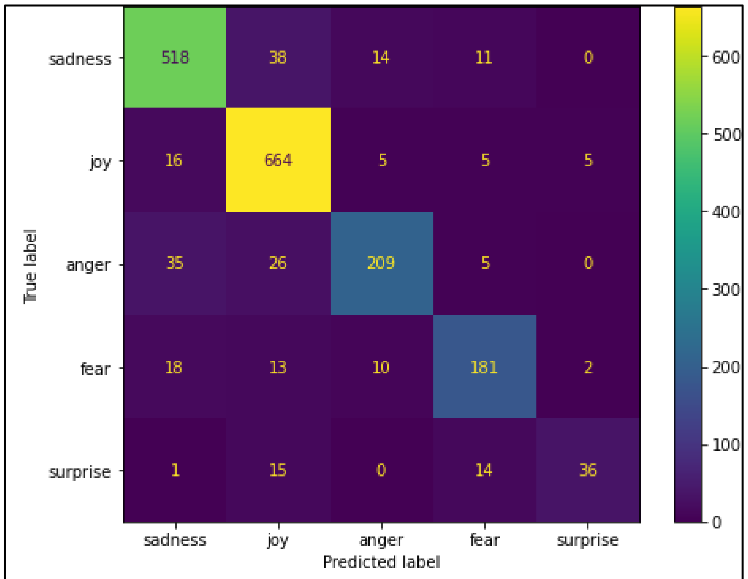
```
              precision    recall  f1-score   support

     sadness       0.84      0.90      0.87       550
         joy       0.87      0.93      0.90       704
       anger       0.88      0.77      0.82       275
        fear       0.83      0.71      0.77       212
    surprise       0.77      0.54      0.64        81

    accuracy                           0.86      1822
   macro avg       0.84      0.77      0.80      1822
weighted avg       0.85      0.86      0.85      1822
```

Figure 15 - 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.



```
              precision    recall  f1-score   support

     sadness       0.88      0.89      0.89       581
         joy       0.88      0.96      0.92       695
       anger       0.88      0.76      0.81       275
        fear       0.84      0.81      0.82       224
    surprise       0.84      0.55      0.66        66

    accuracy                           0.87      1841
   macro avg       0.86      0.79      0.82      1841
weighted avg       0.87      0.87      0.87      1841
```

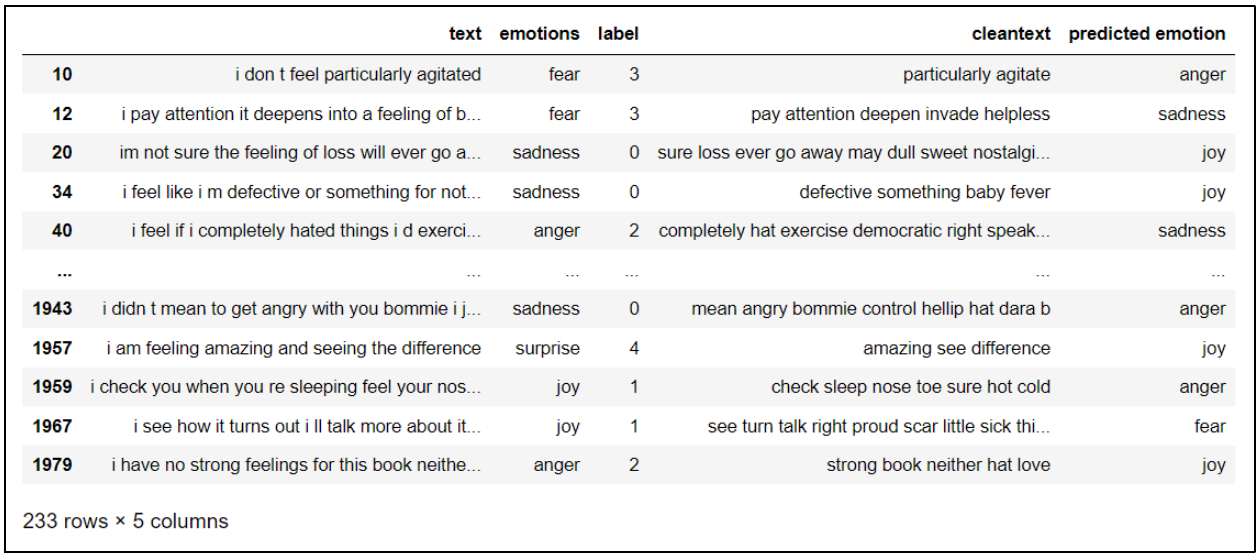
Figure 16 - SVM classification report on testing set

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



	text	emotions	label	cleantext	predicted emotion
10	i don t feel particularly agitated	fear	3	particularly agitate	anger
12	i pay attention it deepens into a feeling of b...	fear	3	pay attention deepen invade helpless	sadness
20	im not sure the feeling of loss will ever go a...	sadness	0	sure loss ever go away may dull sweet nostalgi...	joy
34	i feel like i m defective or something for not...	sadness	0	defective something baby fever	joy
40	i feel if i completely hated things i d exerci...	anger	2	completely hat exercise democratic right speak...	sadness
...	...	...	...	...	...
1943	i didn t mean to get angry with you bommie i j...	sadness	0	mean angry bommie control hellip hat dara b	anger
1957	i am feeling amazing and seeing the difference	surprise	4	amazing see difference	joy
1959	i check you when you re sleeping feel your nos...	joy	1	check sleep nose toe sure hot cold	anger
1967	i see how it turns out i ll talk more about it...	joy	1	see turn talk right proud scar little sick thi...	fear
1979	i have no strong feelings for this book neithe...	anger	2	strong book neither hat love	joy

233 rows × 5 columns

Figure 17 - Preview of wrong predictions from testing set

It appears that there are 233 records that were falsely classified.
Class count of wrong predictions

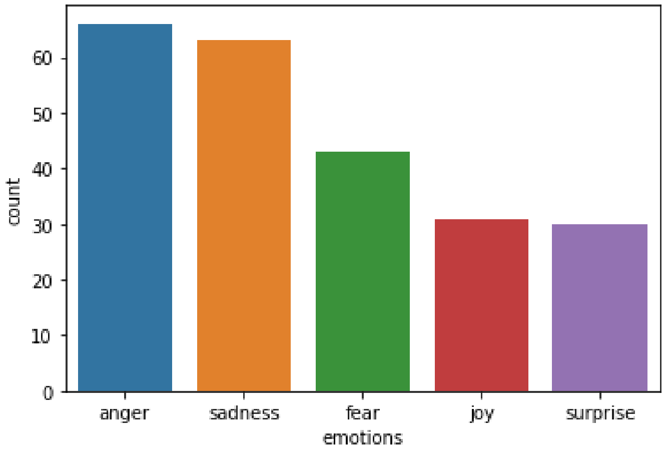


Figure 18 - falsely predicted emotions from testing set class count

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

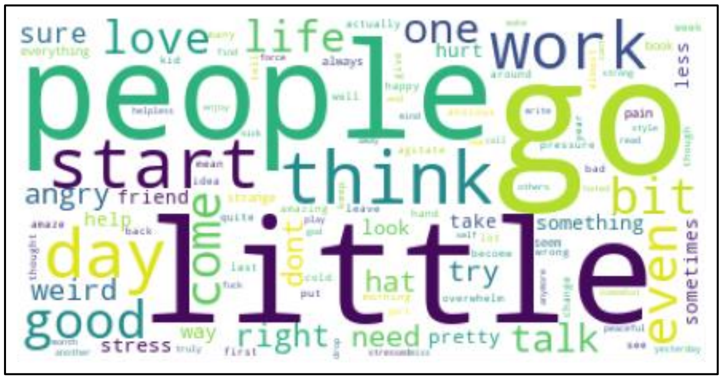


Figure 19 - word cloud for misclassified records of testing set

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"

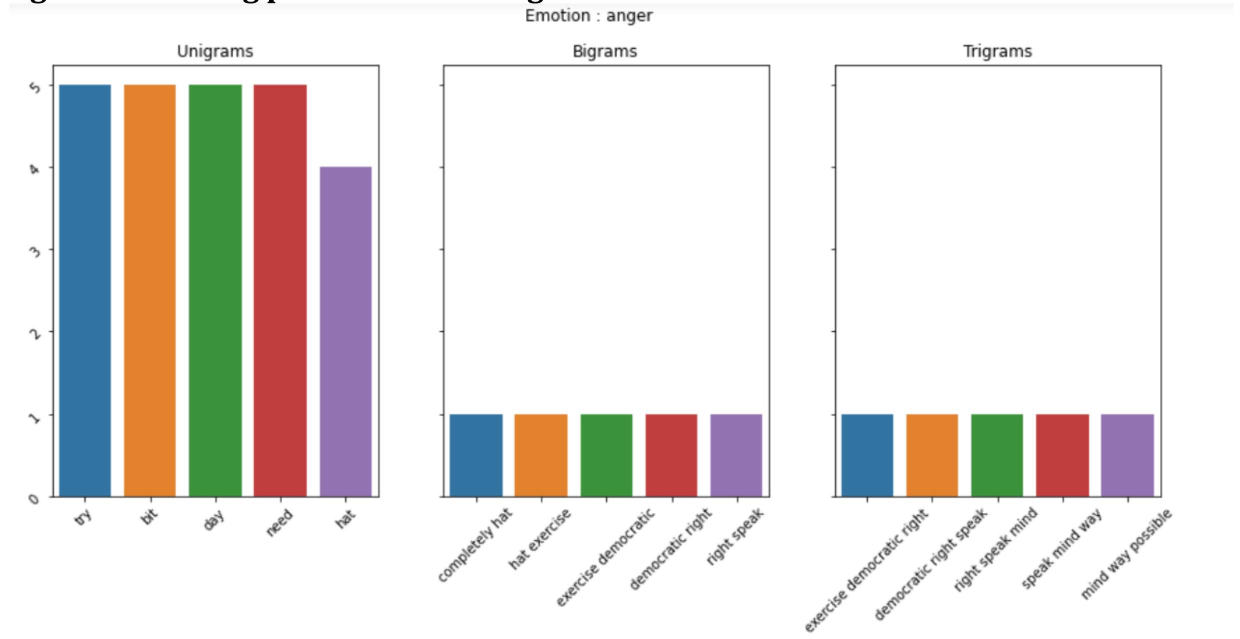


Figure 20 - N-grams of misclassified "anger" emotion text from testing set

N-grams of wrong predictions of "sadness"

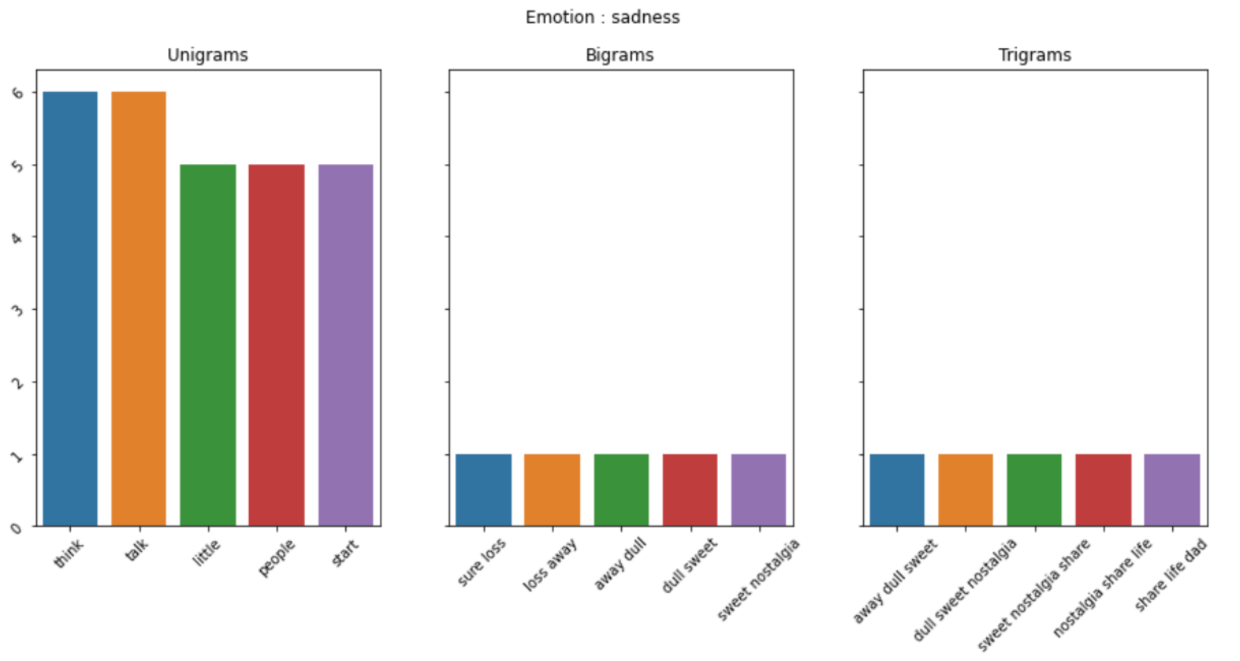


Figure 21 - N-grams of misclassified "sadness" emotion text from testing set

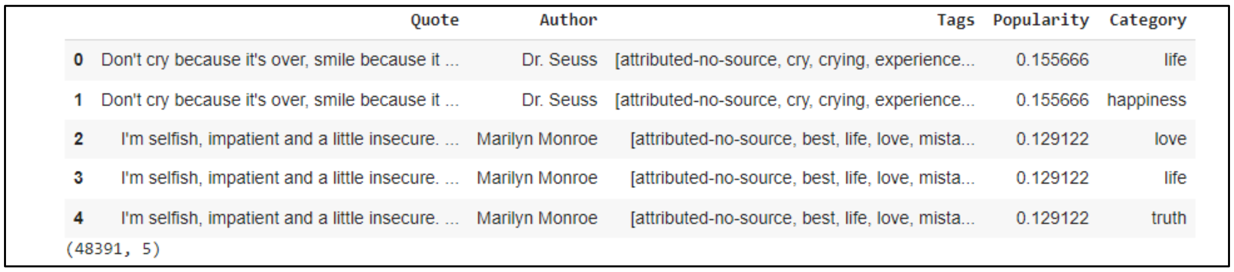
From looking at both n-grams, it can be noticed that most of the unigrams are unbiased and can classified into different emotions, but for the bigrams they show clearly show the emotion of the text, yet their frequency is very low, so these terms were probably ignored because of the configured TF-IDF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

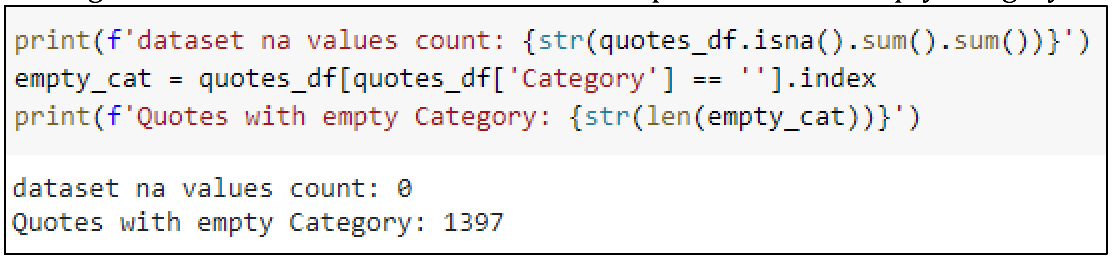


	Quote	Author	Tags	Popularity	Category
0	Don't cry because it's over, smile because it ...	Dr. Seuss	[attributed-no-source, cry, crying, experience...	0.155666	life
1	Don't cry because it's over, smile because it ...	Dr. Seuss	[attributed-no-source, cry, crying, experience...	0.155666	happiness
2	I'm selfish, impatient and a little insecure. ...	Marilyn Monroe	[attributed-no-source, best, life, love, mista...	0.129122	love
3	I'm selfish, impatient and a little insecure. ...	Marilyn Monroe	[attributed-no-source, best, life, love, mista...	0.129122	life
4	I'm selfish, impatient and a little insecure. ...	Marilyn Monroe	[attributed-no-source, best, life, love, mista...	0.129122	truth

```
(48391, 5)
```

Figure 22 - 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.



```
print(f'dataset na values count: {str(quotes_df.isna().sum().sum())}')
empty_cat = quotes_df[quotes_df['Category'] == ''].index
print(f'Quotes with empty Category: {str(len(empty_cat))}')
```

```
dataset na values count: 0
Quotes with empty Category: 1397
```

Figure 23 - Code snippet for finding NA and uncategorized quotes counts

The uncategorized quotes were dropped

Data visualization

Quotes count per category plot

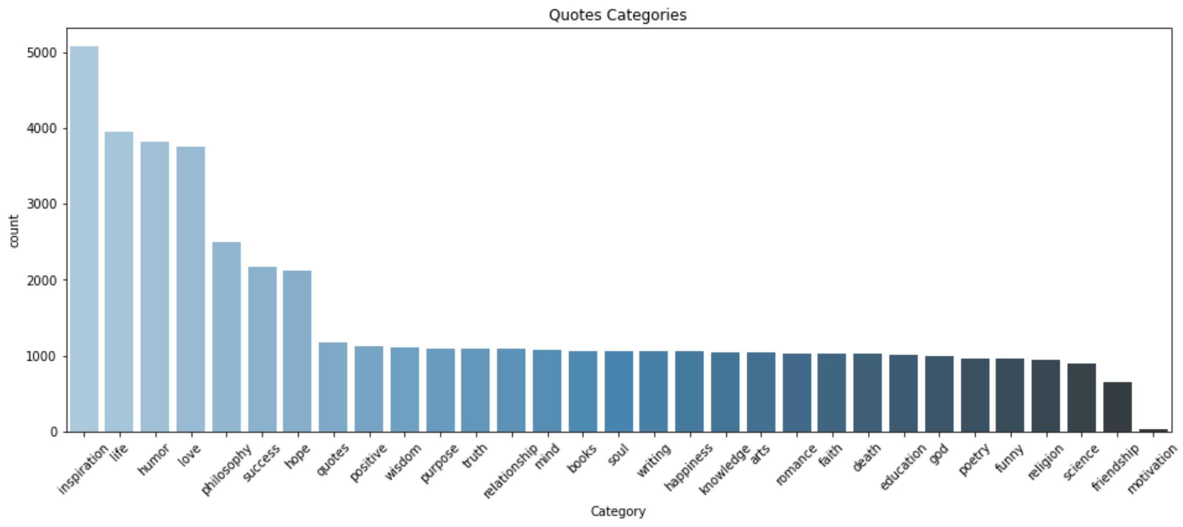
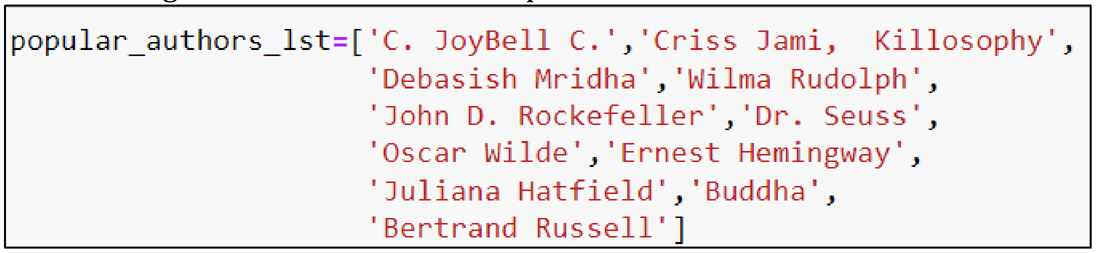


Figure 24 - Number of quotes per category

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.



```
popular_authors_lst=['C. JoyBell C.','Criss Jami,  Killosophy',
                     'Debasish Mridha','Wilma Rudolph',
                     'John D. Rockefeller','Dr. Seuss',
                     'Oscar Wilde','Ernest Hemingway',
                     'Juliana Hatfield','Buddha',
                     'Bertrand Russell']
```

Figure 25 - List of selected Authors

The selected quotes count was 1694.

Selected quotes count per category plot

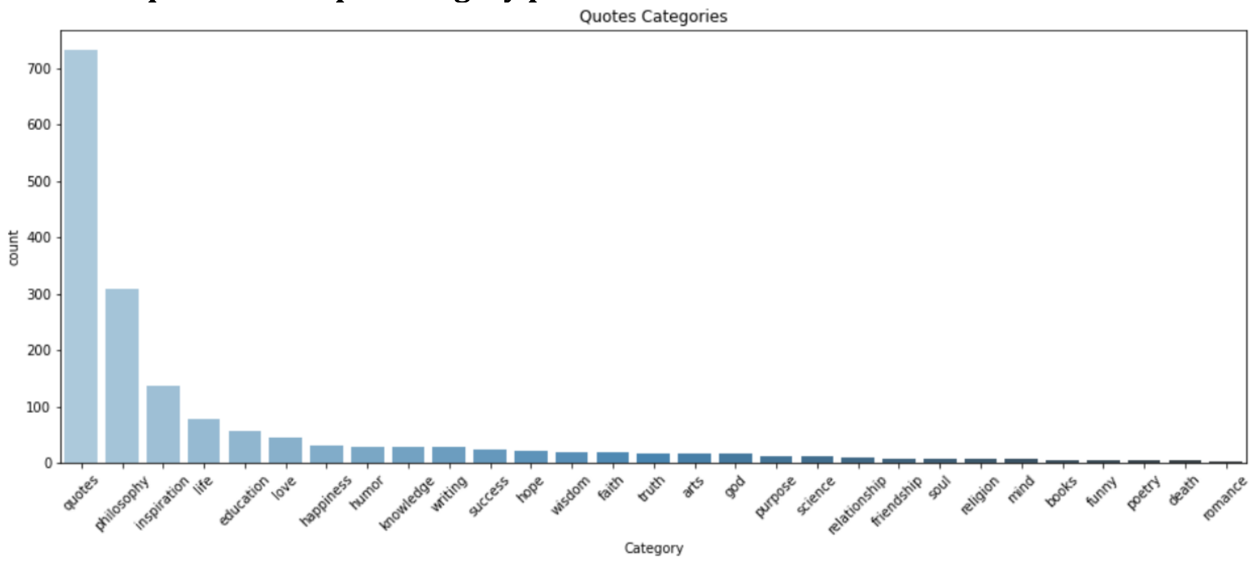


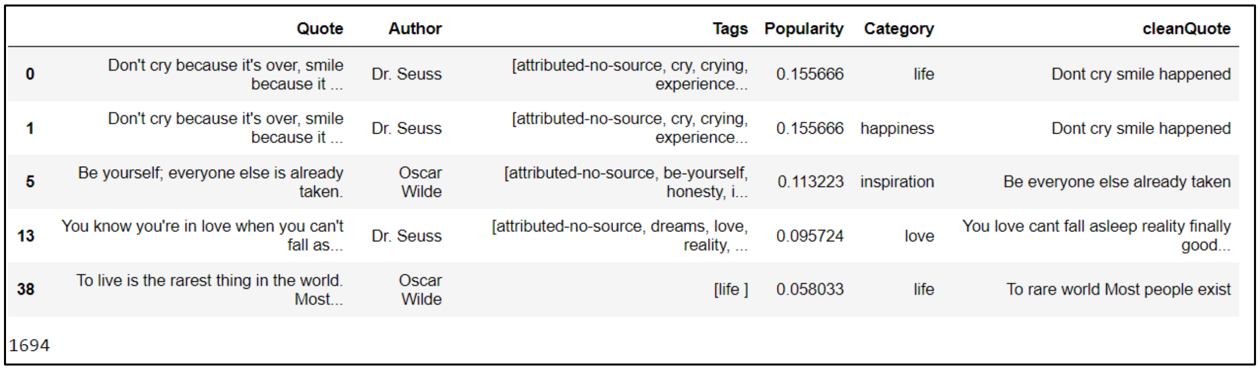
Figure 26 - Number of quotes per category for the selected authors

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.



	Quote	Author	Tags	Popularity	Category	cleanQuote
0	Don't cry because it's over, smile because it ...	Dr. Seuss	[attributed-no-source, cry, crying, experience...	0.155666	life	Dont cry smile happened
1	Don't cry because it's over, smile because it ...	Dr. Seuss	[attributed-no-source, cry, crying, experience...	0.155666	happiness	Dont cry smile happened
5	Be yourself; everyone else is already taken.	Oscar Wilde	[attributed-no-source, be-yourself, honesty, i...	0.113223	inspiration	Be everyone else already taken
13	You know you're in love when you can't fall as...	Dr. Seuss	[attributed-no-source, dreams, love, reality, ...	0.095724	love	You love cant fall asleep reality finally good...
38	To live is the rarest thing in the world. Most...	Oscar Wilde	[life]	0.058033	life	To rare world Most people exist

```
1694
```

Figure 27 - 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.

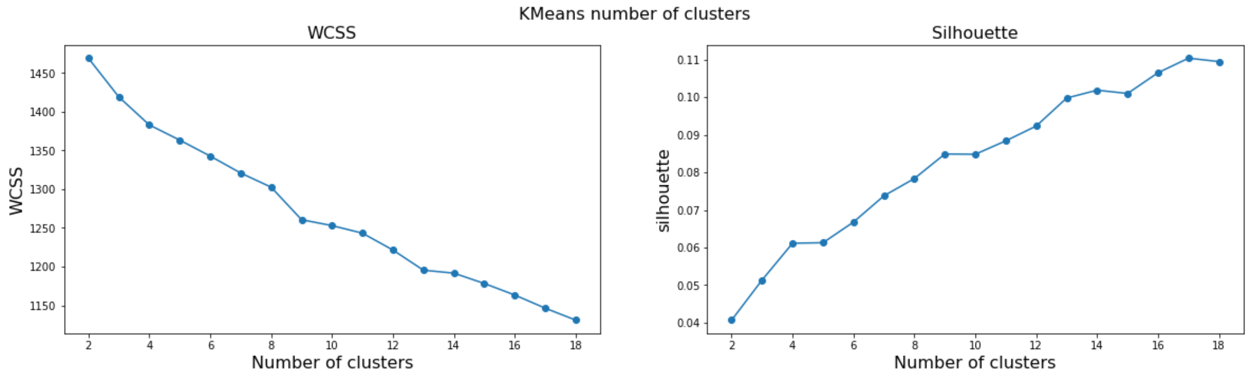
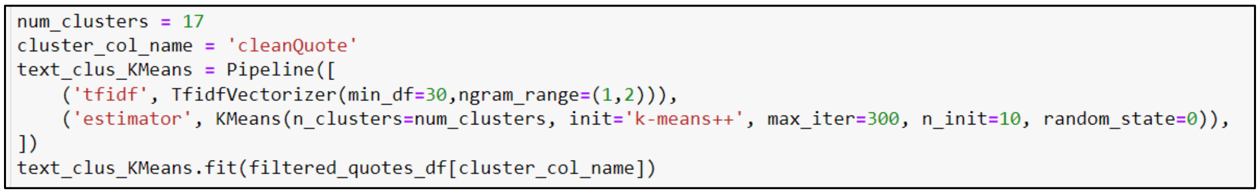


Figure 28 - Plots for selecting the best number of clusters for K-means

The best number of clusters is 17.
A pipeline was created using mentioned TF-IDF configuration and K-Means clustering model as follows:



```
num_clusters = 17
cluster_col_name = 'cleanQuote'
text_clus_KMeans = Pipeline([
    ('tfidf', TfidfVectorizer(min_df=30,ngram_range=(1,2))),
    ('estimator', KMeans(n_clusters=num_clusters, init='k-means++', max_iter=300, n_init=10, random_state=0)),
])
text_clus_KMeans.fit(filtered_quotes_df[cluster_col_name])
```

Figure 29 - 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

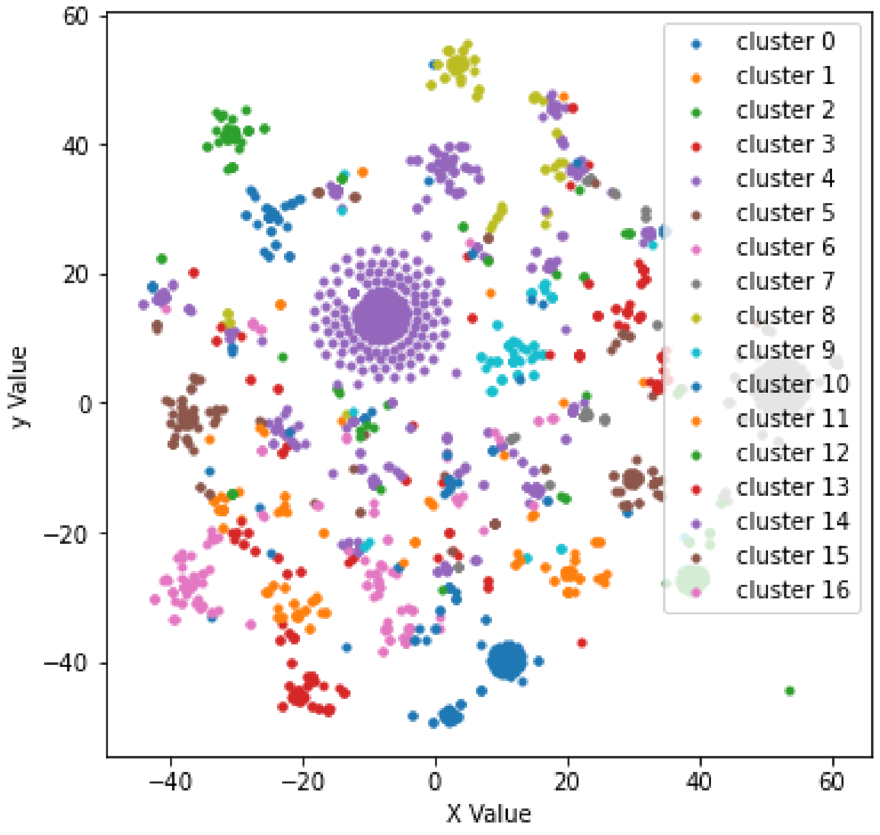


Figure 30 - t-sne plot of K-means clusters

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.

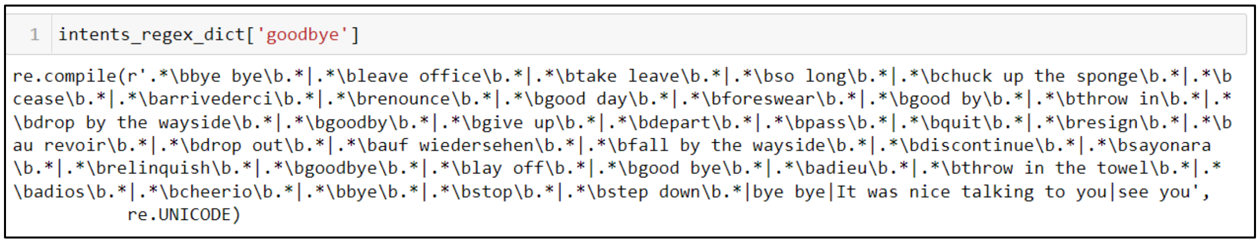
Conversation algorithm

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

Intent matching

Following the suggested method described here 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.



```
intents_regex_dict['goodbye']
```

```
re.compile(r'.*\bbye bye\b.*|.*\bleave office\b.*|.*\btake leave\b.*|.*\bso long\b.*|.*\bchuck up the sponge\b.*|.*\b
cease\b.*|.*\barrivederci\b.*|.*\brenounce\b.*|.*\bgood day\b.*|.*\bforeswear\b.*|.*\bgood by\b.*|.*\bthrow in\b.*|.*
\bdrop by the wayside\b.*|.*\bgoodby\b.*|.*\bgive up\b.*|.*\bdepart\b.*|.*\bpass\b.*|.*\bquit\b.*|.*\bresign\b.*|.*\b
au revoir\b.*|.*\bdrop out\b.*|.*\bauf wiedersehen\b.*|.*\bfall by the wayside\b.*|.*\bdiscontinue\b.*|.*\bsayonara
\b.*|.*\brelinquish\b.*|.*\bgoodbye\b.*|.*\blay off\b.*|.*\bgood bye\b.*|.*\badieu\b.*|.*\bthrow in the towel\b.*|.*
\badios\b.*|.*\bcheerio\b.*|.*\bbye\b.*|.*\bstop\b.*|.*\bstep down\b.*|bye bye|It was nice talking to you|see you',
           re.UNICODE)
```

Figure 31 - 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

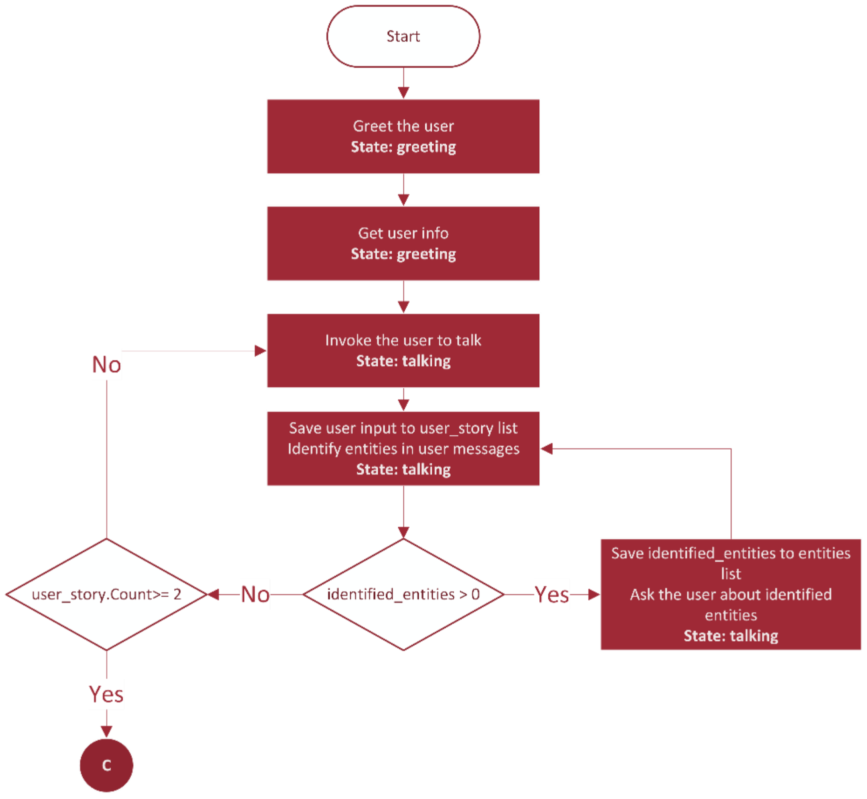


Figure 32 - Flowchart for the "talking" state of the chatbot agent

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:

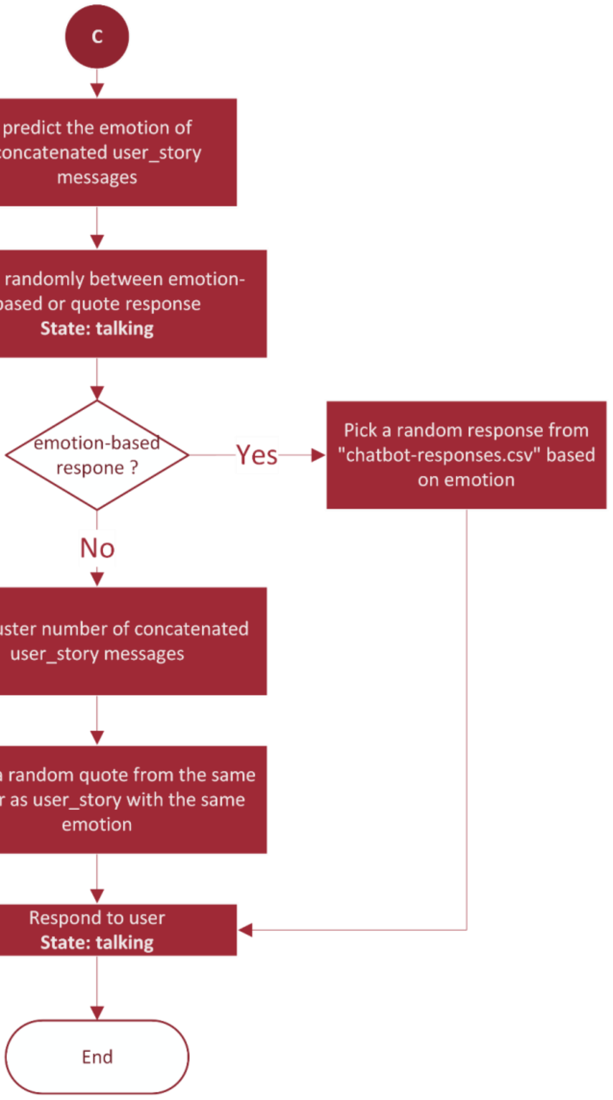

C
predict the emotion of concatenated user_story messages
Select randomly between emotion-based or quote response
State: talking
emotion-based respone ?
Yes
Pick a random response from "chatbot-responses.csv" based on emotion
No
Get cluster number of concatenated user_story messages
Select a random quote from the same cluster as user_story with the same emotion
Respond to user
State: talking
End

Figure 33 - Flowchart for response selection process of the chatbot agent

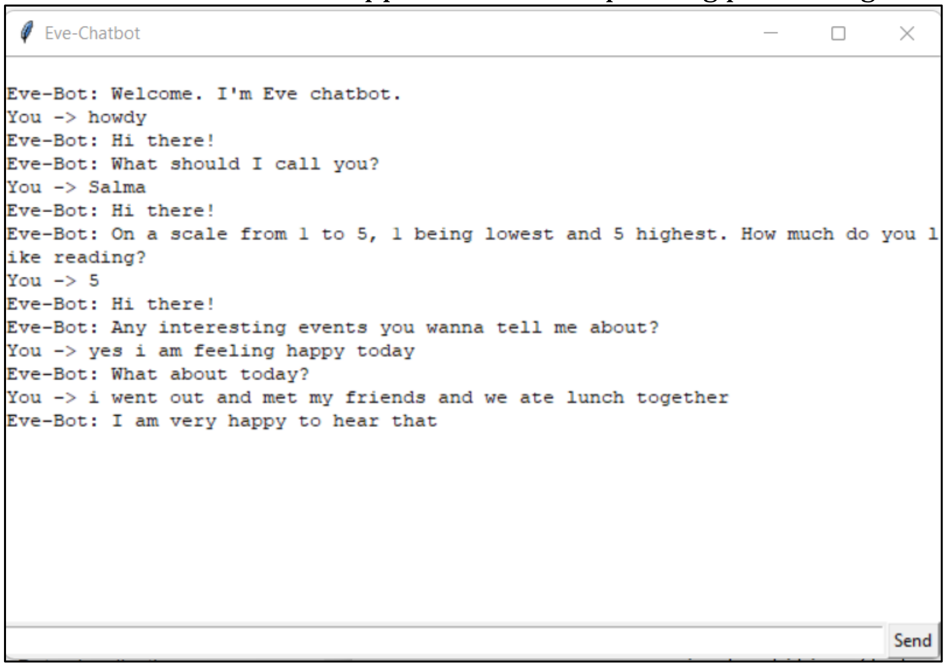
The users answer to liking reading questions is used to increase the probability of responding to the user using a quote, because if the user likes reading, he will probably like being given a quote response. Also, in case the chatbot decides to respond with a quote yet the selected quote cluster does 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.

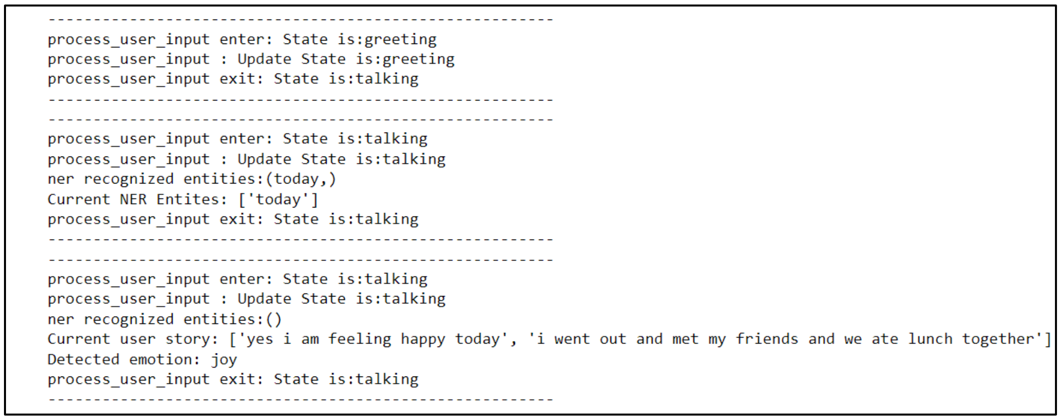
Debugging the conversation flow

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.



```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> howdy
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Salma
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 5
Eve-Bot: Hi there!
Eve-Bot: Any interesting events you wanna tell me about?
You -> yes i am feeling happy today
Eve-Bot: What about today?
You -> i went out and met my friends and we ate lunch together
Eve-Bot: I am very happy to hear that
```

Figure 34 - chatbot conversation example for "joy" emotion



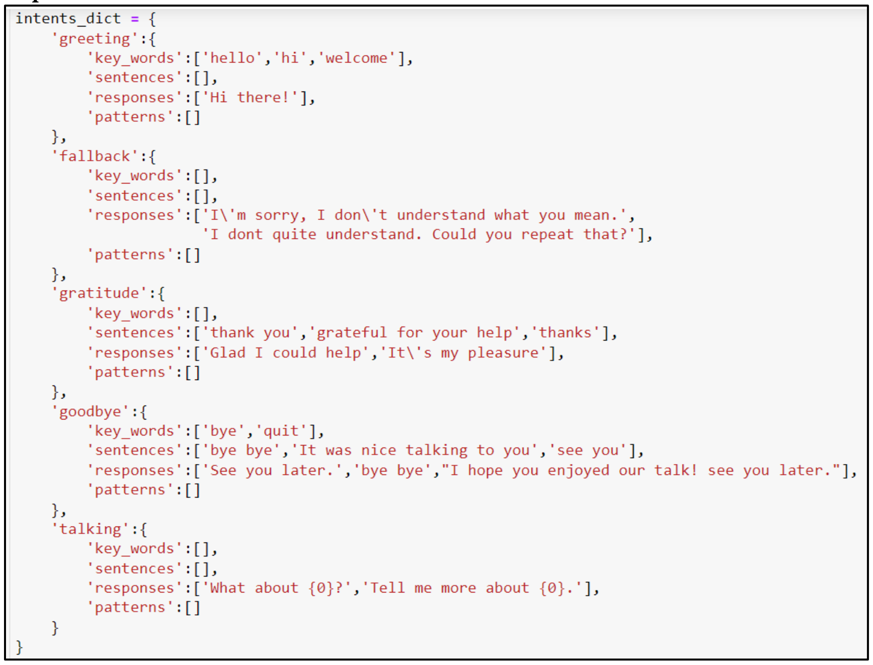
```
--------------------------------------------------------
process_user_input enter: State is:greeting
process_user_input : Update State is:greeting
process_user_input exit: State is:talking
--------------------------------------------------------
--------------------------------------------------------
process_user_input enter: State is:talking
process_user_input : Update State is:talking
ner recognized entities:(today,)
Current NER Entites: ['today']
process_user_input exit: State is:talking
--------------------------------------------------------
--------------------------------------------------------
process_user_input enter: State is:talking
process_user_input : Update State is:talking
ner recognized entities:()
Current user story: ['yes i am feeling happy today', 'i went out and met my friends and we ate lunch together']
Detected emotion: joy
process_user_input exit: State is:talking
--------------------------------------------------------
```

Figure 35 - 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:

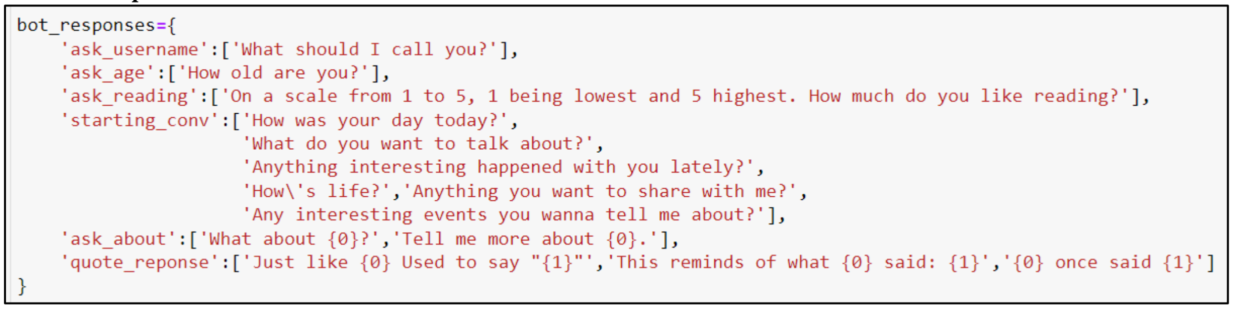
- **Intents dictionary:** a dictionary that has intent name as its key, and the value is a list containing the following items:
    - **key_words:** list of key words that their synonyms are generated, and both the keywords and synonyms are used identify this intent using regex matching.
    - **sentences:** 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.



```
intents_dict = {
    'greeting':{
        'key_words':['hello','hi','welcome'],
        'sentences':[],
        'responses':['Hi there!'],
        'patterns':[]
    },
    'fallback':{
        'key_words':[],
        'sentences':[],
        'responses':['I\'m sorry, I don\'t understand what you mean.',
                     'I dont quite understand. Could you repeat that?'],
        'patterns':[]
    },
    'gratitude':{
        'key_words':[],
        'sentences':['thank you','grateful for your help','thanks'],
        'responses':['Glad I could help','It\'s my pleasure'],
        'patterns':[]
    },
    'goodbye':{
        'key_words':['bye','quit'],
        'sentences':['bye bye','It was nice talking to you','see you'],
        'responses':['See you later.','bye bye',"I hope you enjoyed our talk! see you later."],
        'patterns':[]
    },
    'talking':{
        'key_words':[],
        'sentences':[],
        'responses':['What about {0}?','Tell me more about {0}.'],
        'patterns':[]
    }
}
```

Figure 36 - 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".

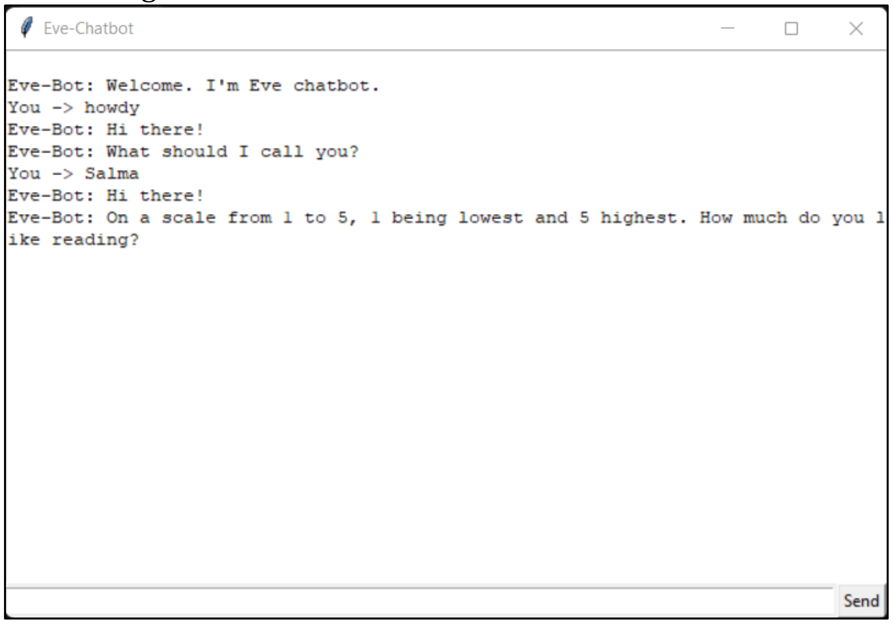


```
bot_responses={
    'ask_username':['What should I call you?'],
    'ask_age':['How old are you?'],
    'ask_reading':['On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you like reading?'],
    'starting_conv':['How was your day today?',
                     'What do you want to talk about?',
                     'Anything interesting happened with you lately?',
                     'How\'s life?','Anything you want to share with me?',
                     'Any interesting events you wanna tell me about?'],
    'ask_about':['What about {0}?','Tell me more about {0}.'],
    'quote_reponse':['Just like {0} Used to say "{1}"','This reminds of what {0} said: {1}','{0} once said {1}']
}
```

Figure 37 - Current static responses for the chatbot agent

GUI

A simple GUI was created using python tkinter library, based on this link. 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.



```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> howdy
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Salma
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
```

Figure 38 - chatbot system GUI

System evaluation

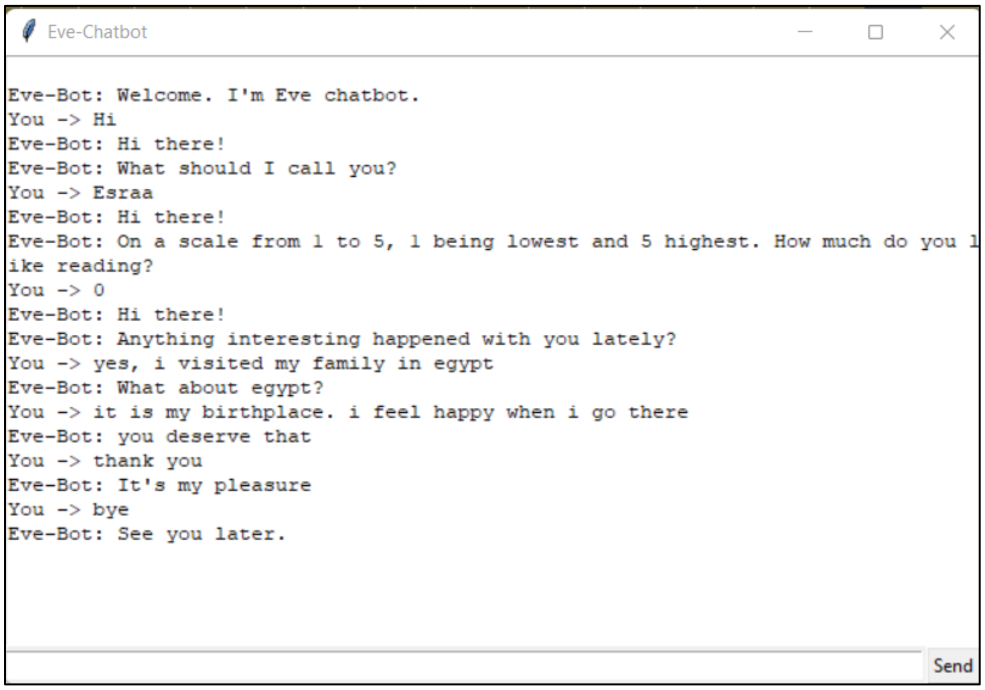
Testing

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

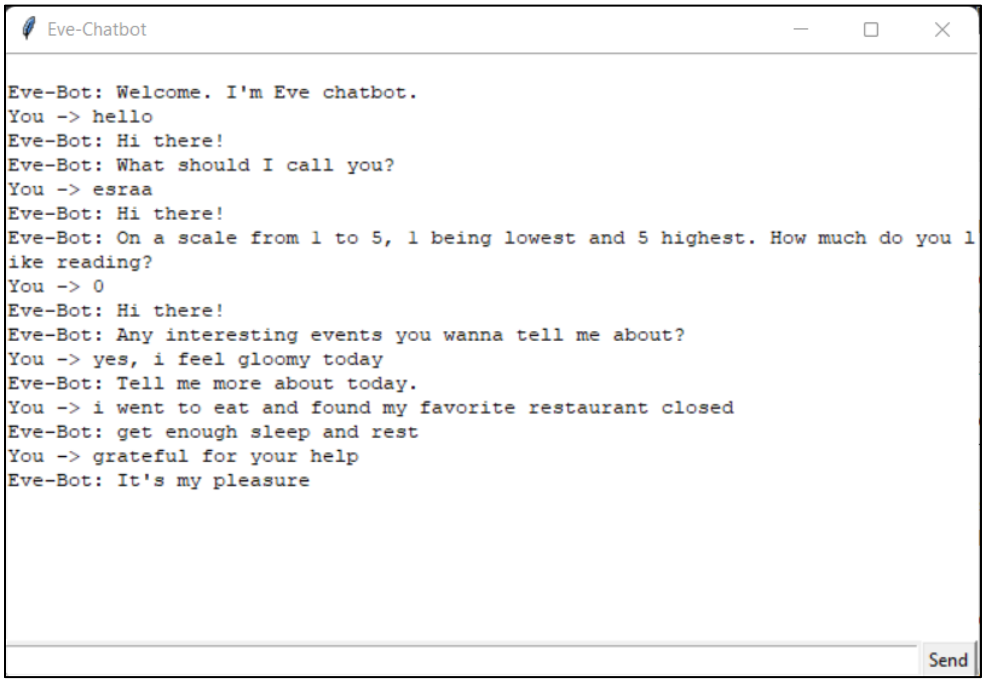


Eve-Chatbot

```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> Hi
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Esraa
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 0
Eve-Bot: Hi there!
Eve-Bot: Anything interesting happened with you lately?
You -> yes, i visited my family in egypt
Eve-Bot: What about egypt?
You -> it is my birthplace. i feel happy when i go there
Eve-Bot: you deserve that
You -> thank you
Eve-Bot: It's my pleasure
You -> bye
Eve-Bot: See you later.
```

Figure 39 - Chatbot conversation example for "joy" emotion

Sadness

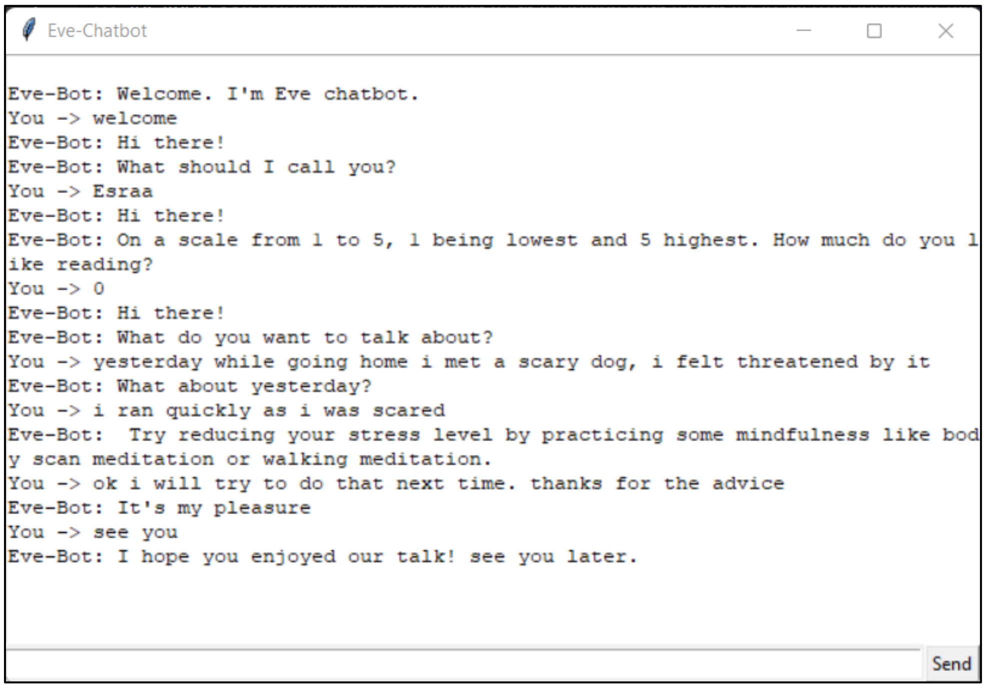


Eve-Chatbot

```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> hello
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> esraa
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 0
Eve-Bot: Hi there!
Eve-Bot: Any interesting events you wanna tell me about?
You -> yes, i feel gloomy today
Eve-Bot: Tell me more about today.
You -> i went to eat and found my favorite restaurant closed
Eve-Bot: get enough sleep and rest
You -> grateful for your help
Eve-Bot: It's my pleasure
```

Figure 40 - Chatbot conversation example for "sadness" emotion

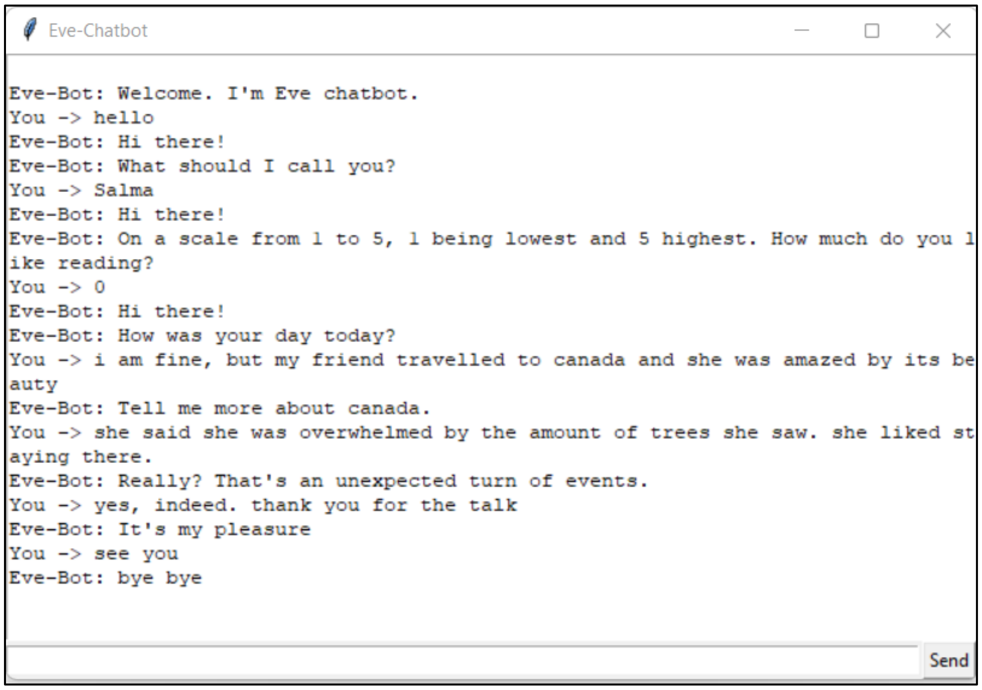
Fear



```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> welcome
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Esraa
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 0
Eve-Bot: Hi there!
Eve-Bot: What do you want to talk about?
You -> yesterday while going home i met a scary dog, i felt threatened by it
Eve-Bot: What about yesterday?
You -> i ran quickly as i was scared
Eve-Bot:  Try reducing your stress level by practicing some mindfulness like bod
y scan meditation or walking meditation.
You -> ok i will try to do that next time. thanks for the advice
Eve-Bot: It's my pleasure
You -> see you
Eve-Bot: I hope you enjoyed our talk! see you later.
```

Figure 41 - Chatbot conversation example for "fear" emotion

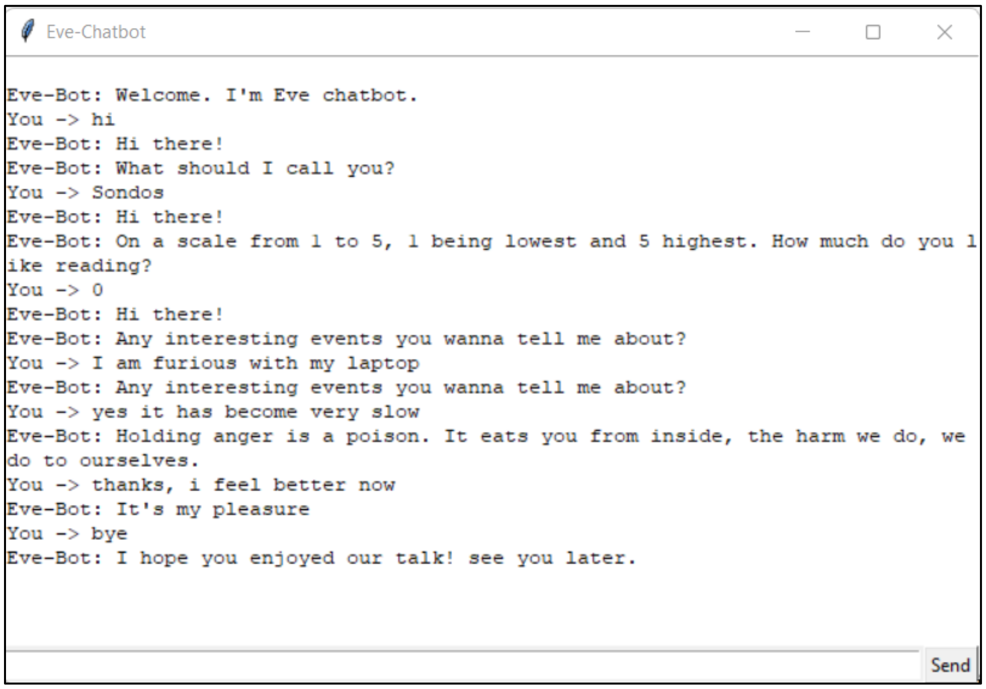
Surprise



```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> hello
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Salma
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 0
Eve-Bot: Hi there!
Eve-Bot: How was your day today?
You -> i am fine, but my friend travelled to canada and she was amazed by its be
auty
Eve-Bot: Tell me more about canada.
You -> she said she was overwhelmed by the amount of trees she saw. she liked st
aying there.
Eve-Bot: Really? That's an unexpected turn of events.
You -> yes, indeed. thank you for the talk
Eve-Bot: It's my pleasure
You -> see you
Eve-Bot: bye bye
```

Figure 42 - Chatbot conversation example for "surprise" emotion

Anger

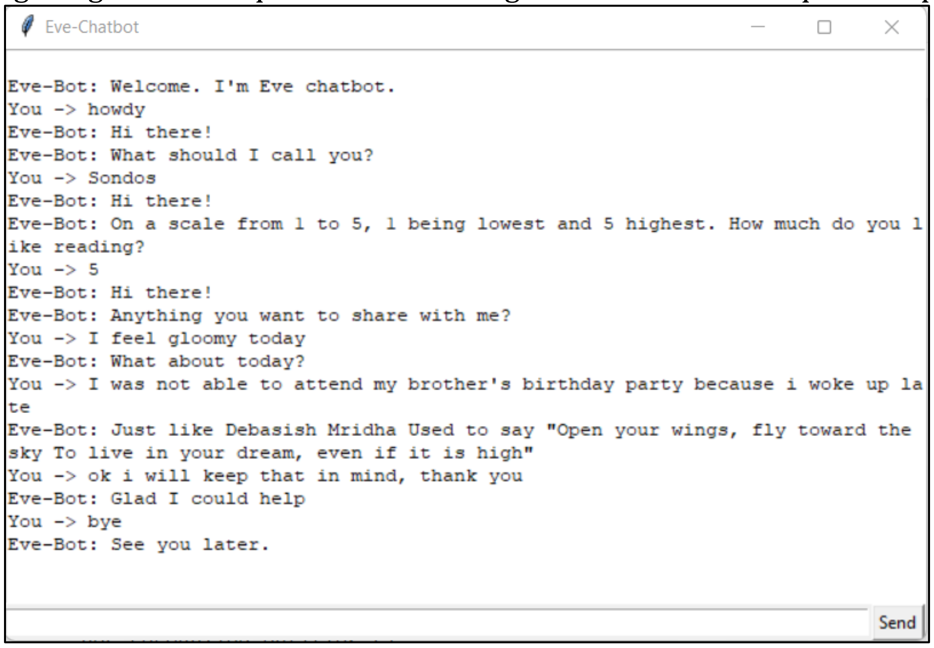


```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> hi
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Sondos
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 0
Eve-Bot: Hi there!
Eve-Bot: Any interesting events you wanna tell me about?
You -> I am furious with my laptop
Eve-Bot: Any interesting events you wanna tell me about?
You -> yes it has become very slow
Eve-Bot: Holding anger is a poison. It eats you from inside, the harm we do, we
do to ourselves.
You -> thanks, i feel better now
Eve-Bot: It's my pleasure
You -> bye
Eve-Bot: I hope you enjoyed our talk! see you later.
```

Figure 43 - Chatbot conversation example for "anger" emotion

Quotes responses testing

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



```
Eve-Bot: Welcome. I'm Eve chatbot.
You -> howdy
Eve-Bot: Hi there!
Eve-Bot: What should I call you?
You -> Sondos
Eve-Bot: Hi there!
Eve-Bot: On a scale from 1 to 5, 1 being lowest and 5 highest. How much do you l
ike reading?
You -> 5
Eve-Bot: Hi there!
Eve-Bot: Anything you want to share with me?
You -> I feel gloomy today
Eve-Bot: What about today?
You -> I was not able to attend my brother's birthday party because i woke up la
te
Eve-Bot: Just like Debasish Mridha Used to say "Open your wings, fly toward the
sky To live in your dream, even if it is high"
You -> ok i will keep that in mind, thank you
Eve-Bot: Glad I could help
You -> bye
Eve-Bot: See you later.
```

Figure 44 - 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.