**EMOTION ANALYSER**

**NATURAL LANGUAGE PROCESSING**

**CSE4022**

**J COMPONENT**

*Submitted by*

*Harshit Lohani*

*16BCE0312*



April 2018

Introduction

Most of the messages sent to you are just plain text. A major problem recognised in these text messages is that we are unable to predict in what emotion is the person texting.

This project deals with a similar situation, as stated above. It takes in a test, and identifies the emotion related to the text. The program takes multimodal input, i.e. it can take input in either text or using your voice. The program includes a speech-to-text Google API that helps it to recognize the words that are said. The program gives output as one of the six basic emotions, which are anger, fear, sadness, disgust, joy and surprise.

[Note: The **dataset** has been taken from another project the link to which is given below

<https://github.com/tpsatish95/emotion-detection-from-text/tree/master/experiments/naive_bayes_classifier> ]

How to install the program:

Pre requisites: The program is made in Python 3.4 and can be run on Python 3.4 and higher version.

The following libraries are required for the main part of the code:

* nltk
* re
* pandas
* defaultdict (imported coollections)

For speech-to-text conversion, we need to download two libraries:

**Python Speech Recognition module:**

to download, type the following command in your command prompt:

sudo pip install SpeechRecognition

**PyAudio:**

Windows users can install pyaudio by executing the following command in a terminal

pip install pyaudio

[Note: Before downloading the two, make sure you have updated the pip file by using the following command:

‘python -m pip install –upgrade pip’]

Step1: :Click on the Github Link given below:

<https://github.com/lohanih/Emotion-Analyser>

From here, either clone the repository or download the zip file and put the “Emotion Analyzer.ipynb” file inside any environment that can run ipynb file with Python3.4+.

Step 2: Adjusting the code according to the work environment:

Once all the libraries are uploaded, open the code editor.

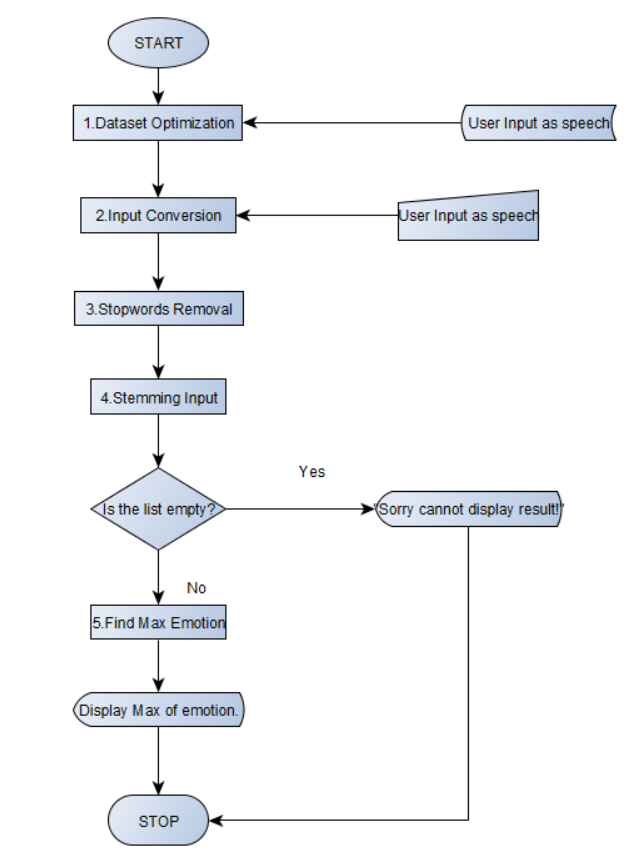
In the line marked as ‘l1’, replace the path given to the path where you have put the dataset, which is provided in the link.

# Step3: Running the code:

Run the code. You must see the Output “Say something”. Speak the sentence that you have to be checked. After some time, it shows the output “You said:” followed by what the code understood from what you said.

The program now shows the result as the emotion of the sentence that you said.

Flowchart

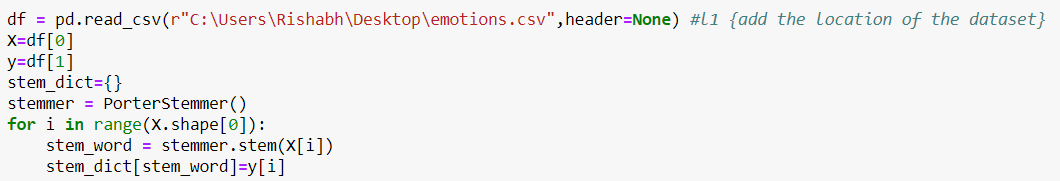


# **Workflow:**

The entire code can be segregated into smaller steps. These steps are:

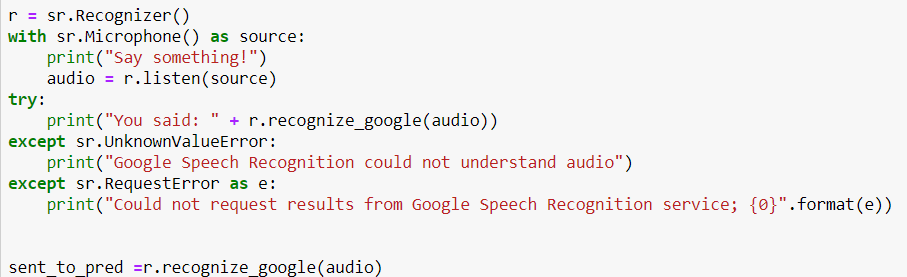
1. **Dataset Optimization:** The dataset file location is put in the given code, two different variables are read for putting the words of the first and the second column in two different variables. The words in the first columns are the various words found in a dictionary and the second columns show the emotion related to their corresponding words. Each words in the first columns are now reduced to their base or root form. The words in first column are put as the key of a dictionary with the values of the second column as their values.

Code:



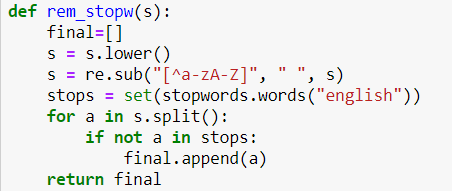
1. **Input Conversion:** In our program we are taking input from the voice of a person. This is converted into a text using a Google speech-to-text API. The API recognizes what you said and then stores it in a variable “sent\_to\_pred”, while simultaneously showing it as an output, to check if the text is what the user has said. In case the program is unable to understand what the user has spoken, it shows an output “Google Speech Recognition could not understand audio” and throws an error.

Code:



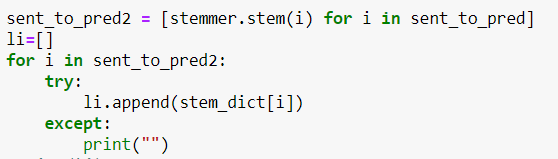
1. **Stopwords Removal:** The String of words generated as an input is sent to a function rem\_stopw(). The function takes the string as an input, changes it all to lowercase, and then removes all the character which are not an alphabet(which do not lie within “a-zorA-Z”). It then takes all the splits the string into the words within the strings and finds the stopwords from the English language (which is downloaded from nltk.corpus) in it. It then appends all the words which are not present as stopwords into a list and return the list from the function.

Code:



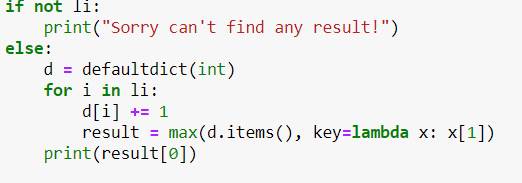
1. **Stemming Input:** In this part of the code, the program first takes the list of word, that it got from the rem\_stopw() function, and stems each and every word present in it so that it can be matched from the dataset.

Code:



1. **Find Max Emotion:** It then finds all the words in this list in the key of the dictionary formed by the dataset. If the word matches the key of any of the dictionary, it adds the corresponding value to another list. Like this all the words are searched in the dictionary and their values are added in the list. If a word is not present is not present in the dictionary, it skips the word and moves to the next word in the list. The list of values of the dictionary is now taken into another dictionary, where keys are all the unique values of the list and the values are the number of occurrences of each of these values. The key with the maximum value is the printed as output, which turns out to be the emotion of the inputted sentence.

Code:



Code Snapshot





Output Snapshot

