######COSC 89

######Assignment 3

######Samuel Lee, David Oh, and Elisabeth Pillsbury

######March 5th, 2018

##Team Emmental: BACchat

Overview:

Given a speech sample, our web application BACchat leverages machine learning and Watson's Speech-to-Text and Tone Analyzer services to determine whether or not an individual is inebriated.

There are two ways to use our application; you may either upload a pre-recorded sample or record yourself live. As BACchat transcribes the audio, it scores it in real time for various emotional, lingual, and social attributes: anger, disgust, fear, joy, sadness, analytical, confident, tentative, agreeableness, conscientiousness, emotional range, extraversion, and openness. BACchat also logs a confidence rating for the transcription of each word.

When the audio clip finishes, BACchat compiles and sends these word confidence ratings and attribute scores to our machine learning module. Finally, BACchat runs this sample through the custom classifier and returns a diagnosis: "You are sober" or "You are drunk".

Business model & motivation:

Although breathalyzers already exist, they are not as readily available as an internet-enabled device. This is where BACchat comes in. BACchat uses cognitive computing to deduce whether a person is drunk or sober by their speech. It is better that we use cognitive computing because it allows us our app to more readily transcribe audio as well as break down the various tones and emotions associated with the transcription. Furthermore, we plan on making BACchat a free service that is readily available via the web to anyone from bartenders to undergraduate college students. We envision this application not only

Knowledge sources:

- Because a large number of machine learning tutorials and support is available in Python, we decided to follow this online tutorial (https://www.digitalocean.com/community/tutorials/how-to-build-a-machine-learning-classifier-in-python-with-scikit-learn) and use Python3 to build our machine learning model and classifier.
- From the beginning, Professor Palmer was very helpful in lending advice on how to approach this project and where to source our audio samples.
- Throughout this process, we regularly referenced StackOverflow (https://stackoverflow.com/) for trouble shooting and help to set up our web application, API, and machine learning classifier.

Tools & techniques:

- We used Scikit-learn (http://scikit-learn.org/stable/) to build our machine learning model and classifier.
- We used Flask (http://flask.pocoo.org/) to set up a simple Python API that transfers our data by bridging the gap between our JavaScript web application and Python3 machine

learning classifier.
- We used Watson's Speech-to-Text (https://www.ibm.com/watson/services/speech-to-text/) service to transcribe audio clips and log confidence ratings for each word transcription.
- We used Watson's Tone Analyzer (https://www.ibm.com/watson/services/tone-analyzer/) service to quantify a text sample in terms of thirteen emotional, lingual, and social attributes.
- We used the Gaussian Naive Bayes machine learning algorithm to train our custom classifier on 60 training audio samples gathered from YouTube (https://www.youtube.com/) and a corpus of sober speech recordings from UCSB (http://www.linguistics.ucsb.edu/research/santa-barbara-corpus#Recordings) ranging in length from 20 to 60 seconds; 30 sample recordings were of sober individuals, and 30 were drunk.

The skeleton of BACchat was built using an open source demo on Github called Real Time Tone Analyzer (https://github.com/IBM-Bluemix/real-time-tone-analysis/), developed by IBM. We made further modifications listed below:

- support uploading pre-recorded audio
- customize frontend styling and layout
- integrate our machine learning classifier using our API
- display final diagnosis

Organization:

- *Data Collection:* David and Elisabeth gathered clips of both drunk and sober individuals, trimmed them to the appropriate length, converted them to WAV files, ran them individually through BACchat to retrieve the word confidence and tone analysis data, and finally, formatted them correctly for classifier training.

- *Machine Learning Classifier, API, and Javascript Integration:* Sam figured out how to create a classifier using Python and Scikit-learn. He organized the training data into the proper file format so that it could be fed into Scikit-learn. He also set up the Python API using Flask and figured out how to send the word confidence and tone analysis data gathered in the frontend through the API to the machine learning classifier.

- *Frontend Web Application:* Elisabeth styled and customized the application's frontend and added file upload functionality.

- *Video:* David (with help from Elisabeth) created the video for submission and presentation.

- *Report:* All three team members wrote this report.

Difficulties encountered:

- *Integrating JS and Python:* We knew that the machine learning portion of this project had to be written in Python. We looked into a couple of options such as spawning a child process from the JavaScript. However, we found that setting up a simple API would be the most straightforward and correct way of integrating the two parts of our application.

- *Sending data from JS to Python:*
We dealt with significant difficulty when we tried to grab the word confidence and tone analysis data and send it to the machine learning classifier.

We wanted the overall tone analysis of the audio clip but the application was configured such that the tone analyzer would request analysis from Watson after every sentence or so. However, after reading through the code, we found that the aggregate of the emotions of the current audio clip could be found. This aggregate was constantly getting updated. Further, there was no indication of when the aggregate was finished getting updated.
To work around this, we would send this aggregate to the Python API as we received it from the tone analyzer. This is written to a file `data.json`. While we never received an end signal when the aggregate was finished updating, we did receive an end signal from the websocket that fires when the speech-to-text terminates. On the websocket closure, we send a GET request to the API to read in the most recent aggregate data in `data.json` and run the machine learning on it.

We also wanted to incorporate the word confidence score from the speech-to-text service. This data also came in every sentence or so. We wanted to incorporate the overall confidence score for the entire audio clip. We did something similar where we send off the data as we get it and the API stores this data in a text file. However, because the JavaScript application does not aggregate this data we must aggregate it on the machine learning side.

Fair assessment of the result:

We were quite surprised that our application was as effective as it was. We started the project quite uncertain about whether using the word confidence scores and the tone values could effectively determine the drunkenness of an individual. In the end, we estimate that our application is currently effective about 70% of the time. We got this estimate by training our personal machine learning classifier with a subset of our data and testing if the remaining data could be correctly classified, which happened around 70% of the time. Of course, there are several steps we could take to improve these results, as mentioned below.

Improvements and further work:

- *Larger Training Set:* Like any machine learning system, more data is typically better. In a future iteration, we would like to collect and train our model on more audio samples. Because we train the model beforehand, a larger dataset would not compromise the speed of BACchat. For reference, training on 60 samples took less than 5 seconds.

- *Display Diagnosis as Confidence Rating:* When classifying an audio sample, the Gaussian Naive Bayes algorithm only returns a binary (`[drunk, sober]`). However, if we implemented a different model, such as a Support Vector Classification, we may be able to return a confidence score. Based on the documentation from the Scikit-learn website, this method would take a longer to train (n^4).

- *Record Real Test Subjects:* For our first stab at BACchat, we sourced our training audio files from YouTube and online speech corpuses. In a future iteration, we would like to gain permission from the IRB to gather interviews from human subjects, both drunk and sober.

- *Improved Speech-to-Text:* The Watson Speech-to-Text service is not robust as we would have liked; if the speaker is muffled or if there is background noise of any sort, the resulting transcription is quite inaccurate. Errors in the speech-to-text portion of our application translate to errors in the tone analysis portion which translate to errors in the sober vs. drunk detection process.

- *Addition of Content Analysis:* As an area of further research, it would be interesting to incorporate semantic content into our machine learning model. Perhaps drunk people talk about food and puppies more often than sober people. Currently, content analysis is not supported within Watson's Tone Analysis service but it would be an interesting avenue to explore in the future.

While BACchat is still in its early stages, we hope that, with the addition of our suggestions above, it may become a viable substitute for breathalyzers in the near future.