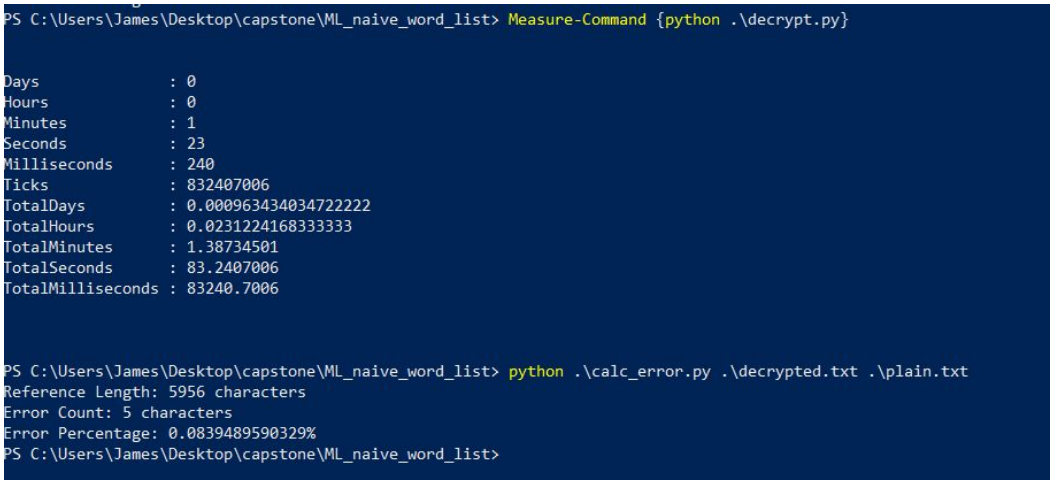
Status Report 2/10/19 - 2/16/19
James Zhang, Team: A2

This past week, I began work on the machine learning portion of the project.

Since we had not begun data collection at the beginning of the week, I started by generating pseudo-data for the cluster-to-key decryption portion of the machine algorithm. This pseudo-data was simply created by taking news articles in English and replacing each letter with another, thus representing a cluster of unknown identity.

In order to begin mapping these clusters back to the correct letter, I first researched English language models. I eventually found GNU Aspell to have good word lists in several languages (https://github.com/en-wl/wordlist). This word list did not, however, include data on word frequency. Thus, I trained the wordlist on plaintexts of novels, updating a frequency counter of each word seen.

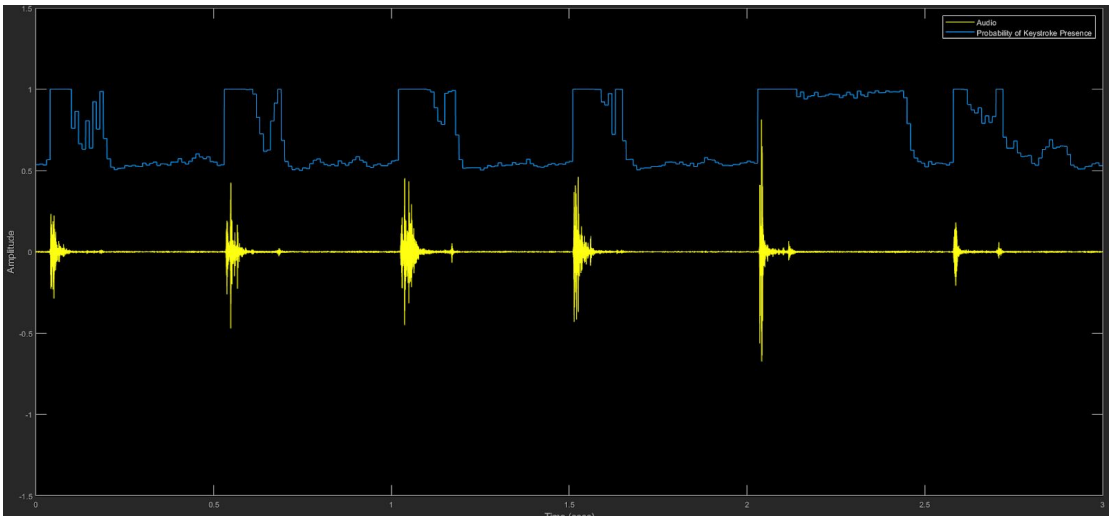
I looked at several different machine learning algorithms to perform this deciphering problem. The first approach was by brute force: mappings between clusters and letters were tried exhaustively until the percentage of misspelled words, based on the word list, was minimized. This method yielded very good character-wise accuracy, at 99.92% (5 wrong characters out of 5956) with a training time of only 1 minute 23 seconds, shown below:



The above approach did not perform well with noise (which may arise from noise interference, poor feature extraction, misclustering, etc.), however. With an introduction of only 5% noise, accuracy fell to 94.50% with a training time of 49 minutes. To account for noise, I implemented a simple spell checker based on the same word list used for decryption. Unfortunately, this spell checker only successfully corrected 28.5% of the incorrect words (61 words out of 214 incorrect). I will later work on refining this spell checker for higher accuracy.

Subsequently, an RNN (using the Keras library with TensorFlow backend) was used. Final accuracy using noiseless data was 98.04%. Training the RNN required significant computation time: each epoch took an average of of around 45 seconds. At 120 epochs, a total time of 94 minutes was required. This is partially due to the fact that the training was performed on the CPU, without AVX fast multiply and accumulate.

I have also begun some preliminary signal processing work using data collected on a cellphone. I have applied a simple bandpass filter to reduce background noise and have implemented rudimentary keystroke detection. The following figure shows the filtered audio (yellow), as well as the probability of keystroke presence (blue).



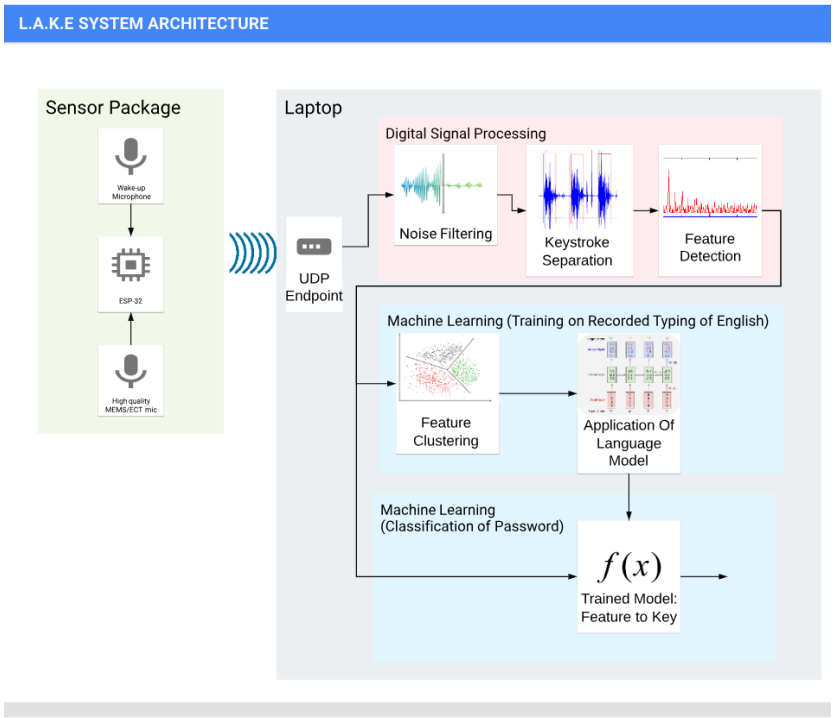
Overall, I have remained mostly on schedule and have achieved my goal of classifying pseudodata successfully using a language model. My goal in the following week is to rebuild TensorFlow for my CPU in order to enable AVX, as attempt to port the code to my GPU. I will then retrain using noisy data and compare with the brute-force attempt. I would also like to look into HMM as another machine learning technique. I will also collaborate with Kevin to help him get the signal processing portion of the system up and running, as well as to work with him to figure out which features (FFT, cepstrum, etc.) I will need to cluster recorded keystrokes.

TEAM STATUS
Team A2

Currently, the most significant risk our team faces is failure to produce a working PCB by the end of the semester. Failure to do so would result in failing to meet our size and power constraints listed for the project. In order to mitigate this risk, we have decided to have Kevin tackle the design of an early prototype of our PCB. We have included additional debug pins and LEDs on the design, which will not be present on our final model, in order to allow us to more easily debug the design. We hope to be able to have this PCB ordered by next week. In the event that we are unable to produce a PCB, our contingency plan would be to breadboard the entire system using breakout a breakout MEMS microphone.

A second concern is the risk of dropping data when sending our data wirelessly. Based on the sampling rate of 44.1kHz, we will be generating approximately 171 kB/s. The ESP32 is limited in memory (512kB of SRAM), so we run the risk of dropping data should we fail to transmit quickly enough. To attempt to mitigate this issue, we will utilize direct memory access (DMA) instead of constantly polling the microphone in order to afford the processor more clock cycles to broadcast data over TCP.

No major changes have been made to the design of our system, and our system architecture will continue to follow the one outlined in our Project Proposal presentation (figured below). Although we did consider use of an accelerometer to help produce additional data, past research has shown that accelerometer based keylogging produced far lower accuracy that acoustic keylogging. The increase in data would also result in increased machine learning training times. Thus, we are currently not planning on introducing an accelerometer into the design.



An updated schedule, which accounts for the earlier implementation of the prototype PCB can be found at the link below:
https://docs.google.com/spreadsheets/d/1zd4GAKGgJ5oQoza22STFgcIHN33uYIu6Kw-OjbusOgg/edit?usp=sharing