Aneem Talukder, Yufei Liu
(NOTE: Brian Hunter was originally in our group, but was dropped because he did not show up for meetings and contributed nothing to this assignment. We already talked to the professor about this)
Spoken Language Processing

Project Part 2 - ASR

How to run our concept recognition program:

We have written a python program called asr.py that takes as its arguements a wave file and a grammar file and it generates a readable table from the results. We have three parallel grammars and therefore it is critical to pick the right grammar for an input for this program.

The three grammars are:
1) gram1.jsgf - this is the grammar for have the system read or query tweets
2) gram2.jsgf - this is the grammar for actions such as favoriting and following users
3) gram3.jsgf - this is the grammar for writing a tweet to Twitter

To run the script, you should first cd into Yufei's directory. Let's say one wants to run the ASR program on a wave file that has a command to read a specific set of tweets. One would have to run a command like the following:

```
cd /proj/speech/users/cs4706/asrhw/yl2515
python asr.py test1.wav 1
```

*Note that the first argument of the program is the wave file for testing and the second argument is a number (the choices are 1, 2, and 3 for gram1, gram2, gram 3, respectively).

Aneem and Yufei have each provided 5 test wav files in the same directory as the python script. The mapping of these files to grammar is as follows:

File	Speaker	Grammar
test1.wav	Aneem	1
test2.wav	Aneem	1
test3.wav	Aneem	1
test4.wav	Aneem	2
test5.wav	Aneem	3
test6.wav	Yufei	1
test7.wav	Yufei	1
test8.wav	Yufei	2
test9.wav	Yufei	3
test10.wav	Yufei	3

Explanation of grammar design decisions:

In order to reduce mistakes that the system could make, we decided to create parallel grammars of subsets of our limited domain. That way we could focus on specific states of our system such as writing a tweet individually. This means that the user will have to interact more with our system, and let it know what his or her intention is. We thought this tradeoff was worth it because we would then be able to work relatively much simpler grammars. Also, this is good because our system has three different states, so it's sensible to design three separate grammars, rather than trying to merge them into a single unmanageable one.

Also, you may notice that all three grammars have "`MAIN MENU`" as a valid sentence, this is a mechanism for our system to go back to the beginning state, regardless of the current state.

For grammar 2, we originally intended to have a single action to be performed every time. However, we realized this was too simple and not a natural way to interact with the system. So we decided to allow 2 parallel actions to be given (although the second action is completely optional). For instance, saying "`I WANT TO FAVORITE THIS TWEET AND FOLLOW THIS USER`" will actually perform two actions at once. Of course, we could go overboard with this idea and allow an unlimited number of actions to be given at once, but that seems rather unnecessary.

Most of our sentences worked well once we added the required words into our custom dictionary (the original dictionary was not nearly complete). However, we did notice the ASR did not perform well when the sentence is short, regardless of which acoustic model we used. So as a fix, we changed those shorter sentences to longer ones. For instance, our grammar 2 originally yielded the sentence "`NEXT`". After having much trouble with it, we changed it to "`NEXT TWEET`" before it was recognized correctly.

Acoustic model that we chose to employ:

We chose to use the 3rd acoustic model. Our reason was simple- it was the only one that did not crash on any of our 10 test files and produces correct recognitions. The others sometimes crashed for Aneem or sometimes for Yufei or even sometimes for both.