**Speech Synthesis**

Noah Brackenbury and Alex Mathson

CS 322, Fall 2015

**Problem**

Many people throughout the world have, by some means or another, lost their ability to produce speech from their vocal chords. While conversing through text is often enough, speech allows us to project to a larger audience and physicalize our thoughts better. Speech can be synthesized through text, however, using analysis of the sounds in our natural languages and analysis of desired text. We created a program that synthesizes speech out of basic English text input.

**Training**

We used recordings of Noah's voice and the Carnegie-Mellon University English-arpabet dictionary to produce speech. After recording all of the possible phones in the arpabet in Noah's voice, we could piece them together in arpabet fashion to form words. We then used CMU's dictionary to convert any given English speech into arpabet.

**Methods**

The first thing we did in this problem is address the smoothing of phones in each word. We found that the best solution we could create was to use frame data to cut the head and tail off of each phone in a word. The algorithm that we used preserves the head of a phone at the beginning and tail at the end of a word, but jams the middle phones against one another to blend the sound.

English words that appear in our CMU dictionary are automatically converted into arpabet to be processed into speech. Any words that are not in the dictionary are redirected to a set of rules that produces a decent but crude guess at the pronunciation of the word.

Improvements

This program can be improved in two major ways. First of all, the smoothing of phones could be less choppy. Using a Fourier transform on the end of one phone on the beginning of the next, and averaging the values for those windows could produce a smoother transition between phones, which would allow the program to sounds better. Secondly, this program could be applied to any topic. Synthesizing speech from the text on a website or app would be a lot more useful than a simple speech synthesis program. We could also have the program train itself on a user's recorded phones to make a personalized speech synthesizer for each user.

Results

The results of this program are hard to quantify, because they depend on a user's understanding of the sound produced. However, it is fair to say that this program produces recognizable speech, especially when the text is also read along with the vocals. Another way to measure the capability of this program is how accurately it produces correct speech from unknown words. Using regular expressions, we were able to create a reasonably good representation of the unknown text in arpabet, and thus, in speech.