Inspiration

Two of my favorite videos on youtube are this one and this one. In both videos, people imitate what English sounds like to non-native speakers. What they are speaking has all of the same sounds as English. They are using all of the vowels and consonants in a native speakers' repertoire, and none of the sounds that English speakers don't use, like the pharyngealized consonants that are in Arabic or the uvular trill that the French use as an r.

What's more, the people in these videos use English sounds *in an English order*. This is because simply stringing together random sounds would not sound like English; you would have impossible vowel polypthongs and consonant clusters. Instead, they use combinations like s-t-r-a-y and a-n-t rather than y-r-t-s-a and a-t-n, following implicit rules. These rules are known to linguistics as Phonotactics, and they vary from one language to another.

I wanted to see if I could do the same thing - create words that sound like English words. I also wanted to explore the probability that certain combinations of words exist in English. I wanted to see what words are the most likely to exist - given English phonotactics. I also wanted to see what combination of sounds would be the most likely to exist in English that was not an actual English word.

Methodology

Rather than model the phonotactics of English using rules, as most linguists do, I thought it would be fun to build a model based on data. To do this, I got phonological data from the Carnegie Mellon Pronouncing Dictionary and a word frequency list. The pronouncing dictionary describes how each of the 113,000 words it contains are pronounced in North American English. For example, the word document is spelled out as D-AA-K-Y-UW-M-EH-N-T. I then mapped the words in the frequency table to their pronunciation in the phonological dictionary, and then built a 2nd-order Markov model, describing the probability of any English sound occurring after other sounds, as well as the probability that a sound would be at the start or end of an English word.

Markov models describe the probability of a system changing between multiple states. These kinds of models are often used in Natural Language Processing, where the system is a sentence, and the states are the word that is about to come next. This is the kind of model your smart phone is using when it suggests words when you are texting. If you type "I think I'll", your phone suggests "be", "have", or "just" as the next word to type because these are very common words to follow that phrase. It doesn't suggest "cogitate" because "I think I'll cogitate" is a phrase that people rarely text to each other. I did something similar, only I used individual sounds (or phonemes) as the unit of analysis, and not words.

I did all of the analysis in R, and because I couldn't find a good package for 2nd-order Markov models, I just build the model myself. It is essentially a three-dimensional array, describing the possibility of phoneme z occurring, given that phonemes x and y were the last two to occur. Because English has 39 unique phoneme and I included a null state, the array is 40 x 40 x 40 dimensions. The scripts for this project are in a github repo here.

Example (maybe add code blocks?)

As an example, lets say we want to generate a string of sounds, where each sound is the most likely to occur given the two sounds before it. The first thing to do is the see which letter is the most likely to start a word. Because my model has a null state, we'll select the 40 x 40 matrix from the three-dimensional array where the first step is null. Then, because we don't have a second state, we will sum one margin of the matrix to get a vector of 40 probabilities, one for each sound in English. For the sounds NG and UH (as in put), the probability is 0, because no English words begin with these sounds. As it happens, the most likely phoneme to begin a word is AH (as in A-bout or bUs), which is the sound that starts 12.5% of all words.

Now that we have two steps (null and AH), we can look at the probability of any subsequent phoneme by selecting the vector from our model where the first dimension is null and the second dimension is AH. This shows that the most likely phoneme to come after a word that starts with AH is N. So far, nothing too surprising - a lot of words begin with the prefix "un-".

Now that we have two letters, we can find the most likely word to follow these two, which is D. To be clear, at this point the model is not counting the conditional probability of an initial null state, so D is the most likely sound to occur after an AH and an N anywhere in a word.

As it happens, the next most likely state is the null state again, because N-D is a very common way for English words to end (ie, bend, mend, and, land, cleaned)

#Results

##Probabilities based on word length

I found the most likely combination of sounds for words from two to six phonemes long, including the probability of the start and end state. While I don't know how to compare probabilities between word lengths, I was able to calculate the probability of *n* sequential sounds in a specific order.

###Two sounds

1 0.DH.AH.0 4.522638e-03 the

2 0.B.IY.0 1.499964e-03 be

3 0.T.UW.0 1.202279e-03 to

4 0.AH.N.0 1.096812e-03 un-

5 0.AH.V.0 1.030409e-03 of

6 0.IH.N.0 8.666870e-04 in

7 0.IH.T.0 1.943523e-04 it

8 0.S.T.0 1.712434e-04 -

9 0.Y.UW.0 1.402617e-04 you

10 0.AE.T.0 1.229933e-04 at

11 0.AO.R.0 1.148554e-04 or

12 0.HH.IY.0 9.733339e-05 he

13 0.AA.N.0 7.446371e-05 on

14 0.D.UW.0 6.365913e-05 do

15 0.AE.Z.0 5.427040e-05 is

The word "ST" could obviously never be an English word, and illustrates the limitations of only having a 2nd order Markov model (as opposed to a 3rd or 4th order model). Because many words begin and end with ST, the model generated it as a plausible word. However, all of the other generated words are actual English words (if you count "un" as a word).

###Three sounds

###Four sounds

###Five sounds

###Six sounds

##Gobbledigook

Maybe embed E-Speak sounds. Compare to random english sounds.

#Caveats

None of the prosody or other variables that are in languages. Indeed, if machines could capture this, Siri would sound a lot more realistic.

Also, it's clear that more than a 2nd order model is needed to truly capture the hidden rules of English Phonotactics. Anyone who has a formal background in phonotactics will tell you that English the placement of phoemes is affected by more than just the two preceeding phonemes, and the flaws in this model definitely demonstrate that.

It would also be cool to look at other languages. Langauges that spell words exactly how they sound (like Spanish) would be especially easy to analyze.