Jake Scheller

Professor Lambert

LIT 3330.001

25 April 2019

Exploring Phonotactics with Computational Linguistics

Computational linguistics, also known as natural language processing, is one of the   
most fascinating areas of active research in computer science. The field, with its undeniably   
sci-fi goal of enabling computers to understand human verbal and written communication, is ultimately contributing to the construction of a truly intelligent artificial intelligence—one that blurs the lines between man and machine. While these lofty promises may be far from reality, current research in natural language processing is constantly producing results that have immediate commercial applications, making computational linguistics a lucrative and in-demand field for researchers. As an example of the utility of using computers to perform linguistics, I have developed a program that uses paradigms of computational linguistics to explore the only-slightly-less fascinating topic of phonotactics. In this paper I will present the program, give background on the computational methods and linguistic topics it encompasses, and propose possible future improvements and avenues of inquiry for me to follow.

Phonotactics is the study of how sounds (phonemes, the atomic unit of meaning in spoken communication) can be combined into syllables. Since syllables are themselves combined to form words, sentences, and more complex verbalizations, phonotactic rules are some of the most fundamental rules of any given language. These rules vary between languages, playing a large part in giving each language a distinctive “sound” that is recognizable even to those who don’t speak the language (Gauvain et al. 1). Phonotactics is interesting because of how algorithmic it is. Unlike many other topics in linguistics, phonotactic rules tend to be easy to define and have few exceptions. This makes them easy to implement in a computer.

I wrote a program that uses phonotactic rules to generate a plausible English nonsense word. This is a toy problem without much practical use, but it effectively demonstrates how phonotactics makes English words sound “English.” The program is written in a programming language called Python, which provides a large collection of tools for computational linguistics. In the past I have used a Python tool called NLTK, the natural language toolkit, for tasks such as corpus analysis, word stemming, and part-of-speech tagging. Because this program is so simple, I did not need to use NLTK. Instead, to test this program I used a tool called python-blick, an implementation of an algorithm called BLICK (Hayes and Wilson) which is described later.

My program builds a word one syllable at a time using a simplified set of phonotactic rules. English syllables are constructed of three parts, the onset, the nucleus, and the coda (Curzan and Adams 82). The onset and the coda are optional and made up of consonants and the nucleus is mandatory and almost always consists of a vowel. As such, every syllable is built off of a vowel, except for a small number of syllabic consonants which can stand on their own as a discrete syllable. In English, the onset can contain up to three consonants and the coda can contain up to five consonants. However there are restrictions on which consonants can be used. For example, there are a very limited number of three-consonant sequences that are valid as a syllable onset in English. The word *strong* uses three consonants /stɹ/ before the vowel /ɑ/. But the non-word *stlong*, which differs in only one consonant is hard to say and is obviously not a valid word. Furthermore, some onsets are only possible before certain values. For example, onsets ending with /j/ are only valid before the vowel /u/ as in the word *fuel*.

To build a syllable, my program first picks a random vowel to be the nucleus. This can be either a monophthong or a diphthong. Next, it randomly chooses an onset of a certain length (measured in phonemes) that is phonotactically allowed to occur before the chosen nucleus. The program stores a list of all possible syllable onsets as well as which vowels they are allowed to precede. Finally it uses a similar method to pick a coda that is allowed to follow both the onset and nucleus. The length of the onset of a syllable is determined by the program taking into account the length of the coda of the previous syllable, as well as whether the syllable is the first or last one in the word.

To build a word, the program simply strings together a random number of syllables between one and five. It chooses one syllable at random to have primary stress, and makes the other syllables have either secondary stress or be unstressed. The method I used to generate results is extremely crude and tends to output bad results a significant portion of the time. To remedy this, I added a final step where the program passes the word to the BLICK algorithm and rejects the word if the algorithm deems it to not be sufficiently “valid.”

The BLICK algorithm is a “phonotactic probability calculator” (Hayes 1) that takes a sequence of phonemes, splits them into syllables, and outputs a number representing how much the sequence deviates from English phonotactic rules. The idea is that a plausible non-word like /blik/ (Chomsky and Halle 101) would be assigned a low score because it demonstrates very little deviation from the rules, while an implausible non-word like /bnik/ would be assigned a much higher score. The algorithm does this by using a large, fixed list of constraints. A constraint is essentially a statement that certain sequences of phonemes with specific properties are “phonotactically worse” (Hayes 6). For example, one constraint states that “it’s bad when a labial sound precedes a nonconsonantal labial sound.” Each constraint is assigned a weight which determines how much a word gets penalized when it violates that constraint. Finally, a word is assigned a value by testing it against all constraints and summing up the weights of all the constraints it violates.

The BLICK algorithm does seem to make the program’s results more “English-like.” The results are output as a space-separated list of phonemes in ARPABET notation (Klautau 2), which maps directly to IPA. The following is an example of 5 words output from the program, as well as their IPA transcriptions and assigned scores.

|  |  |  |
| --- | --- | --- |
| ARPABET | IPA | Score |
| Y EY1 L JH | /jeɪldʒ/ | 0.0484 |
| K AE1 . SH ER0 N T | /kæ ʃɚnt/ | 0.0121 |
| S W IY0 . B OW1 . SH AH0 M P S | /swi: boʊ ʃʌmps/ | 0.0045 |
| G Y UW1 DH | /gju:ð/ | 0.0015 |
| D R AH0 . T ER1 . ZH OW0 N | /drʌ tɚ ʒoʊn/ | 0.0001 |

As you can see, these words are certainly more pronounceable than they would be if the program had just generated a random sequence of phonemes. However, they are not all easy to pronounce. I think there is one obvious thing that would improve the output of my program. First, it should take into account the relative frequency of each phoneme (or phoneme n-gram, or consonant cluster, etc.) and only output phonemes according to their relative frequency. This would be easy to implement using, for example, the data in Kessler and Treiman (296) and I believe would vastly improve the “English-ness” of the words. Studies have shown that statistical relationships between phonemes and syllables is one of the first things infants learn when they are beginning to acquire language (Saffran et al 1927). By ignoring phoneme frequencies, my program throws away a lot of the information that English-speakers use to determine whether a sequence of sounds is an English-like word.

Postscript

You can download my program and instructions for running it from my GitHub profile at <https://github.com/JakeScheller/linguistics-final-project>

Works Cited

Gauvain, Jean-Luc, Abdel Messaoudi, and Holger Schwenk. "Language recognition using phone latices." *Eighth International Conference on Spoken Language Processing*. 2004.

Hayes, Bruce, and Colin Wilson. "A maximum entropy model of phonotactics and phonotactic learning." *Linguistic inquiry* 39.3 (2008): 379-440.

Curzan, Anne, and Michael Adams. *How English works: A linguistic introduction*. 3rd ed., Pearson Education, 2013.

Hayes, Bruce. "BLICK: a phonotactic probability calculator (manual)." (2012).

Chomsky, Noam, and Morris Halle. "Some controversial questions in phonological theory." *Journal of linguistics* 1.2 (1965): 97-138.

Klautau, Aldebaro. "ARPABET and the TIMIT alphabet." (2001).

Kessler, Brett, and Rebecca Treiman. "Syllable structure and the distribution of phonemes in English syllables." *Journal of Memory and language* 37.3 (1997): 295-311.

Saffran, Jenny R., Richard N. Aslin, and Elissa L. Newport. "Statistical learning by 8-month-old infants." *Science* 274.5294 (1996): 1926-1928.