Modeling Phonological Rules in Children's Speech Using Computational Phonology

Parker Stakoff

Carissa Redfield

University of Pennsylvania

Abstract

Using several algorithms to create phonological rules that model any given child's speech, we created a program that can predict how a child will say a given word, based on an inputted data set. It does this by implementing several algorithms to abstract the rules from the data, and then using another algorithm to apply these rules to the word we want to guess. While our program, in some senses, only touches on the complex phonological rules that govern a child's speech, it provides an important framework for future endeavors in computational phonology.

Background

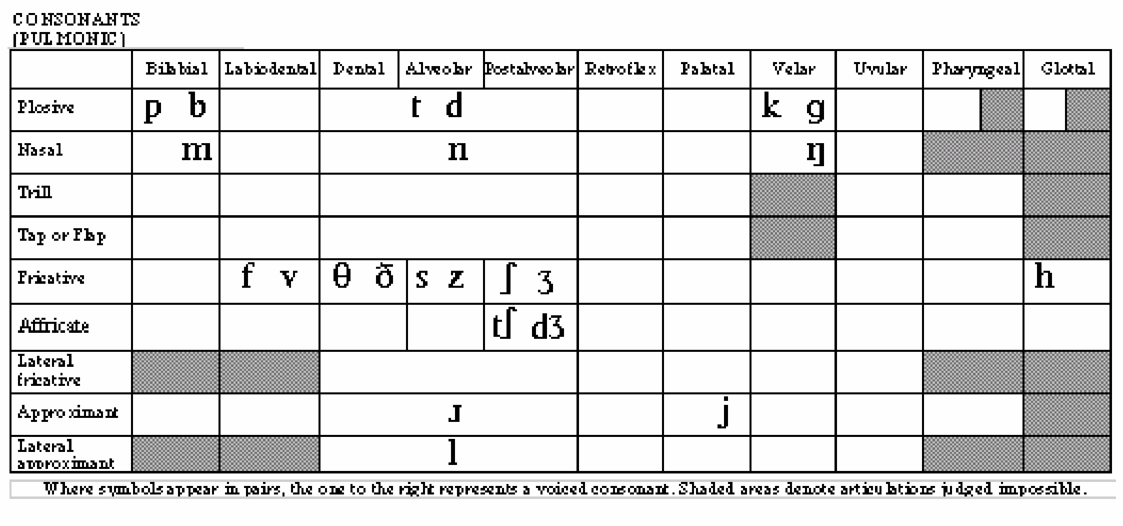
Solving a Phonology Problem

In order to understand how to create a program that properly models the phonological rules in children's speech, we needed to understand how to methodically solve a phonology problem. We used several resources in addition to what we had already learned in the course. Koirala (n.d) and Wassink (2007) emphasize the need to first look at the phonetic environment in which a sound changes and then make a general rule that encompasses all similar sound changes in the phonetic environments in which they occur. We also found that it was helpful to look at each individual sound alternation and figure out if there were alternations that could be encompassed into a larger, more general rule, pertaining to place, manner, and voicing of sounds. We used these observations as inspiration in how we modeled our program.

Representing Linguistic Data in Code

Because the IPA isn't easily accessible in any programming languages that we were familiar with, we used the Arpabet to model the different sounds in English words that would be inputted into the program. Our knowledge of the Arpabet was based off Carnegie Mellon's CPU Pronouncing Dictionary (Lenzo), as well as a Wikipedia article on the symbols it uses.

Moreover, we researched what specific consonants of English we wanted to include, and we based that off of the following chart:



CONSONANTS (PULMONIC)

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d				k g		▨	▨
Nasal	m			n				ŋ		▨	▨
Trill								▨			▨
Tap or Flap								▨			▨
Fricative		f v	θ ð	s z	ʃ ʒ						h
Affricate					tʃ dʒ						
Lateral fricative	▨	▨								▨	▨
Approximant				ɹ			j				▨
Lateral approximant	▨	▨		l						▨	▨

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

Figure 1. IPA Chart for English Consonants (Irwin). A visual chart of the consonants we chose to represent in our phonological rules for the program, not including /w/, which we put in as a voiceless velar approximant.

Any sounds that are not on this chart, but may exist in the Arpabet database could be recognized by our code, but we assume that only these sounds will be in the sound changes that exist for our data set.

Algorithms for Rule-Based Learning

We also looked into what computational phonology projects had been done in the past and were open to the public, in order to understand what kind of algorithms we would implement into our code (1) to create rules from a set of data and (2) to apply these rules to any word we would need to change. We realized pretty early on that there wasn't much inspiration for the first type of algorithm, and created our rule-making algorithms completely from scratch.

Our Project

How it Works

All of the algorithms we used were created first by hand to illustrate the general steps we wanted to achieve, and then modified so they could be implemented in code. The program is given a series of word pairs. These pairs are made of the word that the child is trying to pronounce (Target Word) paired with the word that the child actually says (Actual Word). We break these words down into each of their component sounds (Target Phonemes or Actual Phonemes). We call them phonemes because we assume all of the rules we account for are for changes that are phonemic, not allophonic. For example, we assume that the data would not exhibit a change that says a voiced alveolar stop goes to a voiced alveolar flap, because flaps are not phonemic in English. An illustration of the first algorithm we used is as follows.

```
For every pair of phonemes in the set
If Target Phoneme (TP) and Actual Phoneme (AP) are the same
        For every rule that says TP changes to a phoneme other than itself
                Update the rule so that it no longer applies to TP's phonetic environment
        For every rule that says TP changes to itself
                Update the rule so that it applies to TP's phonetic environment
        If TP maps to no rules
                Create a rule that says TP changes to itself in any phonetic environment
If TP and AP are not the same
        For every rule that says TP changes to itself, or a sound other than AP
                Update the rule so that it no longer applies to TP's phonetic environment
        If there is a rule that already says TP changes to AP
                Update the rule so that it applies to TP's phonetic environment
        If there isn't a rule that already says TP changes to AP
                Construct a rule for the place/manner/voicing of this sound change, and
                make it apply to any phonetic environment
```

Figure 2: Algorithm for Specific (Phoneme to Phoneme) Rule Creation. This figure illustrates the first algorithm we implement in order to create specific rules that map from one phoneme to another. From the rules created by this algorithm, we then create more general rules.

All of the rules created by this algorithm say something like "T changes K in a certain phonetic environment," not something like "place changes from alveolar to velar in a certain phonetic environment." After going through all the data in the training set and creating all these specific phoneme to phoneme rules, we then abstract more general rules from the rules we have already created. For every specific rule that indicates a change in sound (i.e. not a rule like K goes to K, which we do create sometimes in our algorithm), we then abstract the properties of multiple specific rules that are similar in a new, more all-encompassing rule. There are some specific nuances to this. For example, if we're looking at a general rule that encompasses the two alternations S ~ TH and Z ~ DH, and the phonetic environment of S ~ TH says it will never come before a T, in our general rule, we will say that the sound change doesn't occur before T or D. Because consonant clusters tend to mirror the voicing of their surrounding consonants, i.e. Z doesn't come before T, it only comes before D, we are able to say that the sound change won't occur before either sound.

Areas for Improvement

We don't account for alternations that occur after certain properties of vowels (i.e. /t/ → /d/ only after tense vowels or /b/ → /m/ only after nasalized vowels. We did this because vowels are different for speakers all across America, and consonants don't have the same kind of nuanced shifts that vowels are constantly undergoing in different parts of the country. In the same light, we also don't look at vowel changes that occur between target and actual speech.

We were also unable to implement rules that include deletion or insertion, but plan to incorporate those aspects into the program as we develop it in the future.

Significance and Further Applications

While our project is simple and predicts the outcomes of only basic phonological rules, the applications from creating such a program are varied and multi-dimensional. One idea for the future is turning this program into an app, in which parents can input their own children's phonological data, and from that, the app will give suggestions of different vocabulary words parents can employ in order to expose their children to more phonological data and quicken their development. The linguistic data children receive from parents is very minimal, so we predict that more exposure to different sounds in different phonetic environments and different words could help a child improve their speech more quickly. This app could be modified to be used in general areas of speech pathology, especially for parents who can't afford a speech pathologist, but still want to help their children develop. It could also be modified to help with accent reduction, especially if paired with voice recognition software that parsed spoken speech into the Arpabet. This modification of the app could help non-native speakers of English realize phonemic distinctions in English that are allophonic in their native language, and could inform the speaker on how to produce these phonemic sounds.

In the future, we could also make the program as a kind of decoder for children's speech. In this case, the program would receive how a child pronounced a word, and would then output what word the child was trying to say (underlying representation).

We also believe that this program can be modified in order to suit the needs of languages other than English, by modifying the consonantal inventory. There are broader phonological properties for place and manner of articulation that allow for virtually every sound that occurs in spoken languages, that we could use in lieu of the values for our PLACE and MANNER constructors. We catered specifically to how we, in Language Acquisition class, have answered rules regarding phonological changes in children's speech. For example, we have to consonants divided into specific manners (stop v. fricative v. nasal v. affricate, etc.) but we could easily modify the program to include phonological terms and distinctions like continuants v. laterals v. nasals v. stridents instead.

Our program, as it is, is able to predict phonological rules that don't include insertion or deletion with high accuracy, and while testing it with different data, we found that it sometimes was better at guessing what a phoneme would change to than the student who had taken Introduction to Sound Change (Phonology).

References

Arpabet. (2015). In *Wikipedia: The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/wiki/Arpabet.

Irwin, P. (2013). *English consonant articulations* [PDF document]. Retrieved from Canvas course page for Ling 120-001 2013C: https://canvas.upenn.edu/courses/1158918.

Lenzo, K. (n.d.). The CMU Pronouncing Dictionary. Retrieved from http://www.speech.cs.cmu.edu/cgi-bin/cmudict#about.

Koirala, Cesar. *Introduction to Phonology, Day 3* [PDF document]. Retrieved from http://udel.edu/~koirala/phonology/day3.pdf.

Noyer, Rolf. (2015). *Distinctive Feature Charts* [PDF document]. Retrieved from Canvas course page for Ling 230-401 2015A: https://canvas.upenn.edu/courses/1272503.

Tajchman, G., Jurafsky, D., & Fosler, E. (1995). Learning phonological rule probabilities from speech corpora with exploratory computational phonology. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* (pp. 1-8). Association for Computational Linguistics.

Wassinick, A. (2007). *How to Solve a Phonology Problem* [PDF document]. Retrieved from http://faculty.washington.edu/wassink/LING200/lect8_phonology2.pdf.