A Probabilistic Ranking Learner for Phonotactics

Daniel M. Albro

January 6, 2000

1 The Problem – Phonotactic Learning

1.1 Phonotactics Comes Early

One of the first tasks children face in the process of acquiring their first languages is to develop a characterization of which sound sequences are legal or illegal in their native languages. Knowledge of this characterization—phonotactics—develops at or before 10 months of age (Jusczyk *et al.* 1994).

Knowledge they (seem to) have: distributional protocategories (surface inventory), ability to recognize their language, more or less.

Knowledge they don't (seem to) have: morphology, semantics, syntax.

1.2 The Input

1.2.1 Surface forms

- Consists of surface phonetic forms to which the children may or may not have attached a meaning (so we can't assume any sort of semantic bootstrapping, morphemic knowledge, etc).
- We can't assume knowledge of relationships among forms, and therefore we can't assume knowledge of underlying forms.
- This means positive data only—what forms exist—whereas phonotactic learning is also trying to determine what forms don't exist.

1.2.2 Noise

- Most of these phonetic forms represent legal sequences in the target languages, but some do not. These illegal inputs may be due to slips of the tongue, ambient noise, foreign language contact (*e.g.*, foreign language television), or any number of other factors.

1.3 Existing Algorithms

Existing algorithms can be put into three categories:

1. Not robust, and not capable of dealing with phonotactics: Tesar's earlier Constraint Demotion algorithms (Tesar & Smolensky 1993; Tesar 1997) fail (become overly permissive, crash, or fail to terminate) in the presence of noise, and are not designed to deal with the phonotactic learning situation, where underlying forms are unknown.

2. Robust, but not capable of dealing with phonotactics: Boersma's algorithm (Boersma 1997) succeeds in learning OT grammars in the presence of noise, but it does not deal correctly with the phonotactic situation.

3. Not robust, but capable of dealing with phonotactics: The algorithms of Tesar and Prince (Prince & Tesar 1999) and of Hayes (Hayes 1999) can learn phonotactics (some comparisons later), but fail in the presence of noise.

1.4 Goal

The (immediate) goal of this research program is to develop an algorithm that is both robust in the presence of noise and capable of learning constraint rankings relevant to phonotactic knowledge.

Ideally, then, the Probabilistic Error-Driven Phonotactic Ranking Algorithm begins with a set of Optimality Theoretic constraints, divided into faithfulness and phonotactic (markedness) constraints, and assigns probabilities to each pairwise ranking possibility (for example, the probability of some constraint A outranking a constraint B) such that given a set of input data with no noise, the most probable ranking (in fact, the only possible ranking) will accept the smallest possible superset of the input data, given the initial constraints. Given a set of input data with linguistic noise, the space of probable rankings defined by the output of the algorithm is such that any given grammar selected from the ranking space will tend to accept sequences that occurred fairly often in the input data and will tend to reject sequences that were not found with any frequency in the input data.

2 The Algorithm

The basic goal of the algorithm is to determine the probability for each single-pair constraint ranking $A \gg B$ that the ranking is necessary in the grammar accepting the smallest possible subset of the input forms. (In the current implementation these rankings are conceived of as being applied atop a set of base strata in which markedness constraints outrank faithfulness constraints, and faithfulness constraints are ranked such that the least permissive constraints are on top).

2.1 Overall:

1. ("Parameter Estimation"): Learn how to learn phonotactics (find out the likelihood of getting evidence for various ranking pairs, given randomized input grammars).
2. ("Ranking Reinforcement"): Input surface forms and reinforce pairwise rankings found necessary to accept those surface forms.
3. ("Calculation of $P(H|O)$"): Calculate the probability of necessity for each possible pairwise ranking, given the observed inputs from step 2.
4. Return to step 1, if necessary.

2.2 Parameter Estimation:

Repeat ad nauseam:

1. Generate a random input.
2. Generate a random ranking (Keep track for each pair C_1, C_2 of the number of estimation trials with ranking $C_1 \gg C_2$ vs. $C_2 \gg C_1$.)

3. Generate an output by running the input through UCOTP (Albro 1998) with the given ranking.

4. Create the mirror image of that output as an input.

5. Determine a minimally permissive set of pairwise rankings that produces just the expected output.

6. Reinforce (by incrementing a counter) the individual $C_1 \gg C_2$ pairs in the set of rankings produced.

When a minimal number of trials of each pairwise ranking has been run, calculate the Bayesian Estimation parameters as follows:

$$p_{H|H} = \frac{r^e_H}{n^e_H}$$

$$p_H = \frac{r^e_H}{n^e}$$

(H represents a pairwise ranking hypothesis, *e.g.* that $C_1 \gg C_2$ is necessary, $p_{H|H}$ is the probability that the $C_1 \gg C_2$ is found to be necessary by the algorithm when confronted with a random output from a grammar in which $C_1 \gg C_2$ holds, p_H is the probability that the algorithm finds $C_1 \gg C_2$ necessary regardless of whether $C_1 \gg C_2$ holds in the input grammar, r^e_H is the number of estimation reinforcements of H, n^e_H is the number of estimation trials in which ranking H holds in the input grammar, and n^e is the total number of estimation trials).

2.3 Ranking Reinforcement

When the algorithm has performed enough estimation and receives actual surface forms, it follows steps 4–6 of the estimation algorithm for each surface form, thus gathering a value r_H for each pairwise ranking possibility.

2.4 Calculation of $P(H|O)$

Probability of necessity of a pairwise ranking (call this H) given an observed number of reinforcements is calculated via Bayes' Theorem:

$$P(H|O) = \frac{P(O|H)P(H)}{P(O)} = \frac{P(O|H)P(H)}{P(O|H)P(H) + P(O|\overline{H})P(\overline{H})}$$

where $P(H)$ is initially 0.5 (it could be continuously adjusted by Bayesian updating to ease computation, if necessary);

$$P(O|H) = \binom{n}{m} p^m (1-p)^{n-m}$$

(it's a binomial distribution), where p is the value $p_{H|H}$ calculated during estimation, n is the number of observed inputs, and m is the number of reinforcements of the hypothesis (r_H); and

$$P(O|\overline{H}) = \binom{n}{m} \overline{p}^m (1-\overline{p})^{n-m}$$

where $\overline{p} = p_n p_H$, that is, the expected probability of noise (I've used 0.1) p_n times the probability in general of reinforcement of this hypothesis .

Note that the binomial terms $\binom{n}{m}$ cancel out, a fact which makes these equations much more feasible to compute.

2.5 Determination of Strictest Rerankings from a Basic Ranking

1. Take the base ranking (markedness constraints on top, other constraints sorted in order of increasing permissiveness (number of segment time slices allowed by the constraint)) and generate an output from the UR mirror image of the input SR.

2. Use the standard tableau method to compare violations of the expected vs. actual output. This gives a set of reranking pairs which is generally more permissive than necessary

$(C_1 \text{ or } C_2 \text{ or } \ldots \gg C_4 \text{ and } C_5 \text{ and } \ldots)$

3. Apply all of the rerankings to the current ranking. Then generate a new output, and return to step 2, unless the expected output was generated.

4. Sort the resultant rankings $C_i \gg C_j$ by the desirability of C_i as a top ranker (prefer markedness constraints over faithfulness constraints, prefer strict faithfulness constraints over permissive faithfulness constraints).

5. Moving in this order, from the least to the most acceptable rerankings, try to remove each ranking $C_i \gg C_j$ from the set of necessary rankings. This is possible if applying the remaining rankings to the base ranking still produces the expected output (and only the expected output). This step can introduce new rankings, in which case we return to step 3.

3 Case Studies

3.1 The "azba" problem

- The algorithm of Tesar and Prince (Prince & Tesar 1999) (and also that of Hayes (Hayes 1999)) relies on a constraint specificity metric that defines one constraint as more "stringent" as another if it accepts a proper subset of the candidates accepted by the other.

- The algorithm presented here uses a different metric which rates the permissiveness of a constraint by the number of segment time slices the constraint neglects to penalize—this metric is a more direct expression of the idea that grammars should be as strict as possible (constraints are chosen by their raw ability or inability to reduce the size of candidate spaces).

Tesar and Prince provide the following example of where their specificity metric can cause failure to learn the strictest grammar.

Given a language in which voicing is not distinctive in fricatives, but voicing is distinctive in stops, and fricatives can be voiced in regressive assimilation, how do we determine that voiced fricatives are not permitted *except* in regressive assimilations? Example forms: *ba, pa, ab, ap, sa, as, *za, *az, aspa, azba, apsa, *abza, *absa, *apza, *azpa.*

Constraints (using their terminology):

M:AGREE(voi)	Adjacent obstruents agree in voicing
M:*b	Stops must be voiceless
M:*z	Fricatives must be voiceless
F:STOP-VOI/ONS	Preserve stop voicing in surface onsets
F:STOP-VOI	Preserve stop voicing
F:FR-VOI/Ons	Preserve fricative voicing in surface onsets
F:FR-VOI	Preserve fricative voicing

The following inputs to the learner will not pass through the base grammar (which is something like the above in Tesar and Prince's system, where there are two strata) and thus will provide evidence for rerankings: /ba/, which generates [pa]; /ab/, which generates [ap]; and /azba/, which generates [aspa]. /ba/→[pa] is evidence of F:STOP-VOI **or** F:STOP-VOI/ONS $\gg$ M:*b, /ab/→[ap] is evidence of F:STOP-VOI $\gg$ M:*b, and /azba/→[aspa] is evidence of F:STOP-VOI **or** F:STOP-VOI/ONS **or** F:FR-VOI $\gg$ M:*b **and** M:*z. Tesar and Prince's basic algorithm, which ignores constraint permissiveness, chooses F:STOP-VOI in all instances of choice here. Such a choice allows [abza], so it's too permissive a grammar. If their algorithm attempts to account for constraint permissiveness, it can't distinguish between F:STOP-VOI/ONS $\gg$ M:*z and F:FR-VOI $\gg$ M:*z, since neither faithfulness constraint accepts a proper subset of the candidates accepted by the other. However, F:FR-VOI $\gg$ M:*z is too permissive—it allows [z] in all positions. The algorithm presented here correctly chooses F:STOP-VOI/ONS, since it can be shown to apply to a smaller number of time slices, and thus leads to more restrictive grammars.

3.2 Pseudo-Korean

Hayes (1999) illustrates his algorithm using a subset of Korean in which the only permissible surface segments are [d], [t], [t^h], and [a]. Of these, [d] only appears in intervocalic position, and [t^h] never appears word-finally. This state of affairs is accounted for by virtue of the following constraints (Hayes' terminology):

*[-SON, +VOICE]	Obstruents are voiceless by default
*[+VOICE][-VOICE][+VOICE]	*Voiceless segments between voiced ones
*[+SPREAD GLOTTIS]	*Aspiration
*[+VOICE, +SPREAD GLOTTIS]	Voicing is incompatible with aspiration
IDENT(ASP)/___V	Preserve underlying aspiration before vowels
IDENT(ASP)	Preserve underlying aspiration
IDENT(VOICE)/___V	Preserve underlying voicing before vowels
IDENT(VOICE)	Preserve underlying voicing

Given a diet of correct surface forms, Hayes' algorithm comes up with the following ranking: *[+VOICE, +SPREAD GLOTTIS] $\gg$ IDENT(ASP)/___V $\gg$ *[+VOICE][-VOICE][+VOICE], *[+SPREAD GLOTTIS] $\gg$ *[-SON, +VOICE] $\gg$ IDENT(ASP), IDENT(VOICE), IDENT(VOICE)/___V.

The PEDPRA algorithm comes up with the same ranking (modulo differences in notation due to the use of UCOTP, and some additional constraints, as usual for UCOTP). However, when given (extremely rarely, to simulate noise) illegal inputs such as [data] or [at^h], Hayes' algorithm will produce a grammar that promotes IDENT(ASP) above *[+SPREAD GLOTTIS], and IDENT(VOICE)/___V above *[+VOICE][-VOICE][+VOICE] and *[-SON, +VOICE]. In this same situation, the PEDPRA algorithm generates probabilities for rankings that are such that most of the time the ranking matches the one Hayes' algorithm gets with no noise given, but extremely rarely the promotions are possible.

3.3 Diola-Fogny

A first-year phonology problem. Diola-Fogny is a West Atlantic language spoken in Senegal (Sapir 1965). Consonants are:

[p]	[t]	[c]	[k]	
[b]	[d]	[ɟ]	[g]	
[m]	[n]	[ɲ]	[ŋ]	
[f]	[s]			[h]
	[l]			
	[ɹ]			
[w]		[j]		

Essentially all phonemes appear in all positions, there are no consonant clusters of greater than size two, and consonant clusters are of the form *nasal plus nasal* and *nasal plus obstruent*, where the initial nasal is always homorganic to the second. Valid inputs given to the system were as follows: [bɛb], [nigaŋgam], [famfan], [nalalaɲ], [lekuɟaw], [famb], and [namamaɲɟ], which derive from /bɛb/, /nigamgam/, /fanfan/, /nalaɲlaɲ/, /letkuɟaw/, /famb/, and /namaɲɟmaɲɟ/, respectively.

Constraints given to the system included the following (in base grammar order):

*TerneryClusters	No consonant clusters of size greater than 2
*ObsGem	Consonant clusters must contain at least one sonorant
*HeterorganicCluster	Consonant clusters must agree in place
*Struc	Penalize all segments
Ident(place)/_V	Preserve place before surface vowels
Max(C)/_#	Preserve consonants prepausally
Max(C)/_V	Preserve consonants before surface vowels
Max(C)/(V)_(V)	Preserve consonants adjacent to a vowel
Ident(place)	Preserve place
Max(C)	Preserve consonants

From these, the system derives the following rerankings from the base grammar: Ident(place) $\gg$ *Struc, Max(C)/_V $\gg$ *Struc, Max(C)/_# $\gg$ *Struc, Max(C)/(V)_(V) $\gg$ *Struc. Not too exciting, but it covers the generalities...

References

Albro, Daniel M., 1998. Evaluation, implementation, and extension of Primitive Optimality Theory. Master's thesis, UCLA.

Boersma, Paul, 1997. How we learn variation, optionality, and probability. [ROA #221].

Hayes, Bruce P., 1999. Phonological aquisition in optimality theory: the early stages. [Rutgers Optimality Archive #327].

Jusczyk, Peter W., Paul A. Luce, & Jan Charles-Luce. 1994. Infants' sensitivity to phonotactic patterns in the native language. *Journal of Memory and Language* 630–645.

Prince, Alan, & Bruce Tesar, 1999. Learning phonotactic distributions. [Rutgers Optimality Archive #353].

Sapir, J. David. 1965. *A Grammar of Diola-Fogny*.

Tesar, Bruce, 1997. Multi-recursive constraint demotion. [Rutgers Optimality Archive #197].

——, & Paul Smolensky. 1993. The learnability of Optimality Theory: an algorithm and some basic complexity results. Technical report, Department of Computer Science and Institute of Cognitive Science, University of Colorado, Boulder. [Rutgers Optimality Archive #2].