Formal and Computational Approaches to Phonology

Thursday: Maximum entropy phonotactics

Julian Bradfield and James Kirby

University of Edinburgh

Probabilistic phonotactics

Chomsky & Halle (1965) noted that there exist three types of possible phoneme sequences in a language:

1. Those that form existing lexical items (e.g. *brick*);
2. Those that form possible, but non-existing sequences (possible in the sense of being judged by native speakers as well-formed *?blick*);
3. Those judged by native speakers as impossible (**bnick*).

Probabilistic phonotactics

However, not all possible words (sometimes called *accidental gaps*) are judged to be equally well-formed:
E.g., *pring* $\gg$ *shlimp* $\gg$ *fnood* $\gg$ *ʒpʊk* (Scholes, 1966)

Probabilistic phonotactics

However, not all possible words (sometimes called *accidental gaps*) are judged to be equally well-formed:
E.g., *pring* $\gg$ *shlimp* $\gg$ *fnood* $\gg$ *ʒpʊk* (Scholes, 1966)

Thus we might want a theory of phonotactics that captures this gradience in well-formedness. One way to incorporate gradience is to have a *probabilistic* model; and since such judgements are rarely precisely repeatable, and vary from speaker to speaker, a degree of random variation seems a reasonable feature.

Probabilistic phonotactics

However, not all possible words (sometimes called *accidental gaps*) are judged to be equally well-formed:
E.g., *pring* $\gg$ *shlimp* $\gg$ *fnood* $\gg$ *ʒpʊk* (Scholes, 1966)

Thus we might want a theory of phonotactics that captures this gradience in well-formedness. One way to incorporate gradience is to have a *probabilistic* model; and since such judgements are rarely precisely repeatable, and vary from speaker to speaker, a degree of random variation seems a reasonable feature.

We might also like a theory of how phonotactic grammars can be learned from surface forms.

Maximum entropy

A maximum entropy (maxent, log-linear) framework provides a natural way to express generalisations about phonotactics. Logistic regression, PCFGs, Harmonic (OT) grammars, and HMMs are all types of MaxEnt models.

Maximum entropy

A maximum entropy (maxent, log-linear) framework provides a natural way to express generalisations about phonotactics. Logistic regression, PCFGs, Harmonic (OT) grammars, and HMMs are all types of MaxEnt models.

The goal of a MaxEnt model is to *maximize the probability of the data* while *maximizing the entropy of the model*: in other words, to include as much information as is known from the data while making no additional assumptions ('the least biased estimate possible on the given information' – E. T. Jaynes)

Maximum entropy

A maximum entropy (maxent, log-linear) framework provides a natural way to express generalisations about phonotactics. Logistic regression, PCFGs, Harmonic (OT) grammars, and HMMs are all types of MaxEnt models.

The goal of a MaxEnt model is to *maximize the probability of the data* while *maximizing the entropy of the model*: in other words, to include as much information as is known from the data while making no additional assumptions ('the least biased estimate possible on the given information' – E. T. Jaynes)

Entropy is a measure of the amount of random variability in a probability distribution: $-\sum_i P(i) \ln P(i)$.

Maximum entropy

Given an empirical probability distribution P, MaxEnt provides a means of constructing an estimate P^* of that distribution. In particular, the MaxEnt estimate can be shown to be the one closest to P in terms of Kullback–Leibler divergence.

The MaxEnt approach has the distinct advantage of distinguishing the structure of the model and the objective function to be optimized from the method used to carry out the optimization.

KL divergence measures how unlikely P is assuming P^*:
$D_{\mathrm{KL}}(P\|P^*) = \sum_i P(i)\ln(P(i)/P^*(i))$.

Maximum entropy

The general form of a MaxEnt model is as follows.

Maximum entropy

The general form of a MaxEnt model is as follows. Consider a random process which produces an output $y \in Y$ influenced by some context $x \in X$.

Maximum entropy

The general form of a MaxEnt model is as follows. Consider a random process which produces an output $y \in Y$ influenced by some context $x \in X$.

We posit m real-valued *features* (not phon!) to pairs (y, x), with values $f_i(y, x)$.

Maximum entropy

The general form of a MaxEnt model is as follows. Consider a random process which produces an output $y \in Y$ influenced by some context $x \in X$.

We posit m real-valued *features* (not phon!) to pairs (y, x), with values $f_i(y, x)$.

We give a *weight* w_i to each feature.

Maximum entropy

The general form of a MaxEnt model is as follows. Consider a random process which produces an output $y \in Y$ influenced by some context $x \in X$.
We posit m real-valued *features* (not phon!) to pairs (y, x), with values $f_i(y, x)$.
We give a *weight* w_i to each feature.
Then the MaxEnt estimate of the probability of any given outcome $y \in Y(x)$ is then

$$P(y|x) = \frac{1}{Z(x)} \exp \sum_{i=1}^{m} w_i f_i(y, x)$$

where $Z(x)$ is the normalization factor

$$\sum_{y \in Y(x)} \exp \sum_{i=1}^{m} w_i f_i(y, x)$$

Maximum entropy

The general form of a MaxEnt model is as follows. Consider a random process which produces an output $y \in Y$ influenced by some context $x \in X$.
We posit m real-valued *features* (not phon!) to pairs (y, x), with values $f_i(y, x)$.
We give a *weight* w_i to each feature.
Then the MaxEnt estimate of the probability of any given outcome $y \in Y(x)$ is then

$$P(y|x) = \frac{1}{Z(x)} \exp \sum_{i=1}^{m} w_i f_i(y, x)$$

where $Z(x)$ is the normalization factor

$$\sum_{y \in Y(x)} \exp \sum_{i=1}^{m} w_i f_i(y, x)$$

In other words, the log probability of y given x is proportional to a linear combination of m feature values.

Learning MaxEnt weights

Given a model with m features and a set of n *observations*, we then want to find the weight w_m for each feature f_m which maximizes the model's log-likelihood:

$$L(P) = \sum_{x,y} P(x, y) \log P(y|x)$$

Learning MaxEnt weights

Given a model with m features and a set of n *observations*, we then want to find the weight w_m for each feature f_m which maximizes the model's log-likelihood:

$$L(P) = \sum_{x,y} P(x, y) \log P(y|x)$$

Selecting an optimal model under this regime is generally acknowledged to be Hard; there are several optimization methods available, but we won't concern ourselves with the details here.

Applications in phonology

There have been multiple applications of MaxEnt to phonological learning problems: Keller (2000, 2006), Goldwater & Johnson (2003), Jäger (2004) and others.

Hayes & Wilson (2008) tackle the problem of learning a phonotactic grammar. Their approach is especially interesting because they propose a method to learn both constraints and rankings directly from surface forms.

The core idea: *well-formedness can be interpreted as probability.*

MaxEnt for phonotactics

In Hayes and Wilson's model, features are OT-style (markedness) constraints, each of which is associated with a nonnegative, real-valued weight.

In this respect it resembles Harmonic Grammar (Smolensky, 1986; Smolensky & Legendre 2006) or Keller's (2000, 2006) Linear OT, as opposed to 'classical' OT, where constraints are strictly dominated.

However, it's worth noting that attempts at capturing gradience have also been made in standard OT, e.g. Boersma & Hayes (2001).

MaxEnt for phonotactics

Hayes and Wilson decompose the MaxEnt equation into three parts, termed a *score*, a *maxent value* and a *probability*.

The *score* $h(x)$ of a phonological input form x is

$$h(x) = \sum_{i=1}^{N} w_i C_i(x),$$

where w_i is the weight of constraint C_i and $C_i(x)$ is the number of times form x violates constraint C_i. (This is analogous to the feature functions described earlier.)

MaxEnt for phonotactics

Hayes and Wilson decompose the MaxEnt equation into three parts, termed a *score*, a *maxent value* and a *probability*.

The *maxent value* $P^*(x)$ of a phonological input form x is

$$\begin{aligned} P^*(x) &= \exp(-h(x)) \\ &= \exp\Big(-\sum_{i=1}^{N} w_i C_i(x)\Big), \end{aligned}$$

in other words, the conditional probability given earlier, without the normalizing constant. (The score is negated so that forms with more violations get lower values.)

MaxEnt for phonotactics

Hayes and Wilson decompose the MaxEnt equation into three parts, termed a *score*, a *maxent value* and a *probability*.

The *probability* $P(x)$ of a phonological input form x and its maxent value $P^*(x)$ is then

$$P(x) = P^*(x)/Z$$

where $Z = \sum_{y \in \Omega} P^*(y)$ is the normalizer.

Learning constraints and weights

Learning constraint weights, given a set of constraints, is an optimization problem we won't discuss in depth here. (H&W use an iterated hill-climbing search procedure; other methods are possible. See Malouf (2002) for an overview.)

Learning constraints and weights

Learning constraint weights, given a set of constraints, is an optimization problem we won't discuss in depth here. (H&W use an iterated hill-climbing search procedure; other methods are possible. See Malouf (2002) for an overview.)

An (arguably more interesting) problem is determining the set of constraints, given the input data. Given the huge number of distributional generalizations consistent with any given surface form, this problem is similarly non-trivial...

Learning constraints and weights

While often modellers assume the constraint set is given by UG, H&W back off and assume instead that the learner has access to

1. a set of (universal) distinctive features;
2. the inventory of segments in the target language;
3. the feature specifications for each of those segments.

(Where do *they* come from? See e.g. Mielke 2004, Lin 2005...)

Learning constraints

In particular, H&W posit constraints that target *natural classes* of features, but these are basically just functions that take as input a sequence of feature matrices, and return a number of matches.

Learning constraints

In particular, H&W posit constraints that target *natural classes* of features, but these are basically just functions that take as input a sequence of feature matrices, and return a number of matches.

So for instance a constraint such as *[+son, +dors] penalizes the segment [ŋ] (et al.); a constraint like *[+cons] [+cons, +cont] penalizes a sequence of C + fricative (e.g. *df, pθ, sh*); and so on.

Learning constraints

In particular, H&W posit constraints that target *natural classes* of features, but these are basically just functions that take as input a sequence of feature matrices, and return a number of matches.

So for instance a constraint such as *[+son, +dors] penalizes the segment [ŋ] (et al.); a constraint like *[+cons] [+cons, +cont] penalizes a sequence of C + fricative (e.g. *df, pθ, sh*); and so on.

They also permit constraints to have at most one *negated* feature matrix: e.g. [αF][^βG] means 'feature F with value α cannot be followed by feature G with value β'.

Learning constraints

In particular, H&W posit constraints that target *natural classes* of features, but these are basically just functions that take as input a sequence of feature matrices, and return a number of matches.

So for instance a constraint such as *[+son, +dors] penalizes the segment [ŋ] (et al.); a constraint like *[+cons] [+cons, +cont] penalizes a sequence of C + fricative (e.g. *df, pθ, sh*); and so on.

They also permit constraints to have at most one *negated* feature matrix: e.g. [αF][^βG] means 'feature F with value α cannot be followed by feature G with value β'.

This allows concise expression of *logical implication*: *[αF][^βG] therefore means [αF] must be followed by [βG].

Learning constraints

If C is the number of natural classes (combinations of features) for a given set of features, and n is the number of feature matrices that may occur in a given constraint, it is clear that the number of possible constraints can grow quite large even for smallish values of n and C (say, $n = 4$ and $C = 100...$)

Learning constraints

If C is the number of natural classes (combinations of features) for a given set of features, and n is the number of feature matrices that may occur in a given constraint, it is clear that the number of possible constraints can grow quite large even for smallish values of n and C (say, $n = 4$ and $C = 100...$)

H&W keep this manageable by employing underspecified feature representations to reduce the size of C and holding n at 2 (for segmental constraints).

Learning constraints

If C is the number of natural classes (combinations of features) for a given set of features, and n is the number of feature matrices that may occur in a given constraint, it is clear that the number of possible constraints can grow quite large even for smallish values of n and C (say, $n = 4$ and $C = 100...$)

H&W keep this manageable by employing underspecified feature representations to reduce the size of C and holding n at 2 (for segmental constraints).

Even with a reasonable number of constraints (possibly in the tens of millions, but not higher), search heuristics are necessary to identify the most accurate and general constraints in the set.

Learning constraints

H&W employ two heuristics: *accuracy* and *generality*.

Accuracy is assessed as the number of observed violations of a constraint divided by the expected number of violations given the current grammar

Learning constraints

H&W employ two heuristics: *accuracy* and *generality*.

Accuracy is assessed as the number of observed violations of a constraint divided by the expected number of violations given the current grammar

Constraints are then sorted using a stepwise accuracy scale (actually a statistical upper confidence limit on O/E, s.t. a difference obtains between a constraint of accuracy $0/10$ and one of $0/1000$).

Learning constraints

H&W employ two heuristics: *accuracy* and *generality*.

Accuracy is assessed as the number of observed violations of a constraint divided by the expected number of violations given the current grammar

Constraints are then sorted using a stepwise accuracy scale (actually a statistical upper confidence limit on O/E, s.t. a difference obtains between a constraint of accuracy $0/10$ and one of $0/1000$).

Generality is assessed by length (shorter constraints $\gg$ longer constraints) and number of segments covered by the natural classes the constraint expresses.

Learning the grammar

Algorithm alternates between selection and weighting: a (general, accurate) constraint is selected from the universal set, the weights are recomputed, another constraint is selected, etc.

Learning the grammar

Algorithm alternates between selection and weighting: a (general, accurate) constraint is selected from the universal set, the weights are recomputed, another constraint is selected, etc.

Phonotactic learning algorithm
Input: a set Σ of segments classified by a set F of features, a set D of surface forms drawn from Σ^*, an ascending set A of accuracy levels, and a maximum constraint size N

Learning the grammar

Algorithm alternates between selection and weighting: a (general, accurate) constraint is selected from the universal set, the weights are recomputed, another constraint is selected, etc.

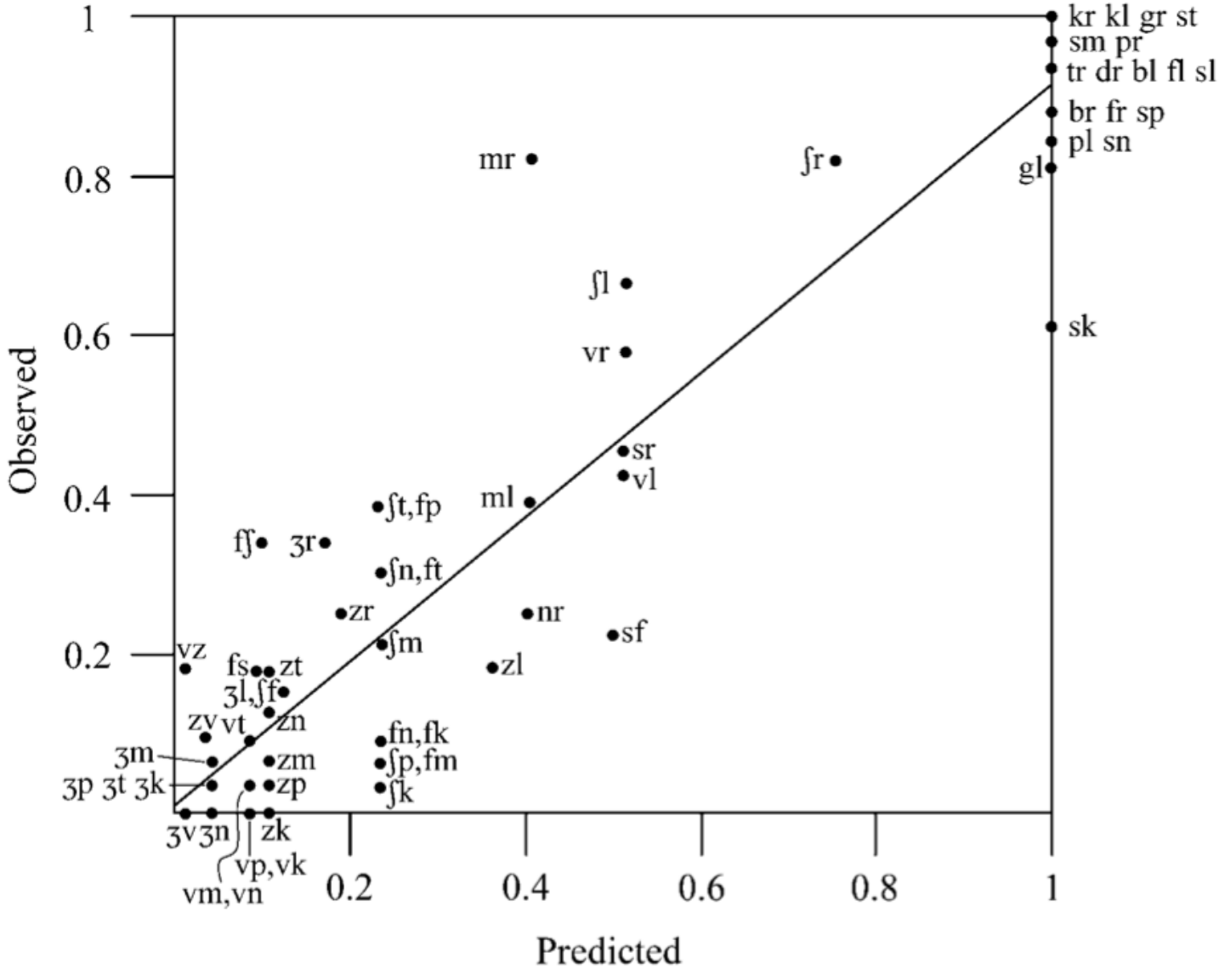
Phonotactic learning algorithm
Input: a set Σ of segments classified by a set F of features, a set D of surface forms drawn from Σ^*, an ascending set A of accuracy levels, and a maximum constraint size N

1 begin with an empty grammar G
2 for each accuracy level a in A do
3 repeat
4 select the most general constraint with accuracy less than a (if one exists) and add it to G
5 train the weights of the constraints in G
6 until no constraint is selected in step 4

So how does it do?

Demo: English onsets

So how does it do?



More complex phonotactic relations: vowel harmony

Shona: five vowel system [i e a o u] with asymmetric height harmony

i $\rightarrow$ e / {e, o} ____ (so *bibiʒa* but *bebera*, **bebira*)

u $\rightarrow$ o / o ____ (so *baduka* but *bodoka*, **boduka*)

[o] occurs noninitially only if preceding vowel is also [o]; [e] occurs noninitially only if preceding vowel is [e] or [o]

(Conversely, the above rules say [u] occurs noninitially only if preceding vowel **isn't** [o], and [i] occurs noninitially only if preceding vowel **isn't** [e] or [o].)

[a] is freely distributed.

More complex phonotactic relations: vowel harmony

If there is a heuristic/practical limit on the gram size, how does the learner deal with nonlocal phonotactics like vowel harmony?

Answer: poorly.

	Constraint	Weight	Comment
1.	*[−back][ ][−high, −low, +back]	4.20	*[e,i][ ]o
2.	*[+low][ ][−high, −low, +back]	1.35	*a[ ]o
3.	*[+high][ ][−high, −low]	3.77	*[i,u][ ][e,o]
4.	*[−high, −low][ ][+high, −low, −back]	3.19	*[e,o][ ][i]
5.	*[+low][ ][−high, −low]	4.15	*a[ ][e,o]

- Correctly prohibits e.g. *?momina*, **memina*, **mamemo*...
- but incorrectly permits *?momuma*, **monduma*, **mendima*

More complex phonotactic relations: vowel harmony

Solution: allow constraint induction over a subset of the representation: the **vowel tier** (or *projection*)

[−seg]	g	o	n	d	w	a	[−seg]	*Default projection*
[−seg]		$\begin{bmatrix} +\text{high} \\ -\text{low} \\ +\text{back} \\ +\text{seg} \end{bmatrix}$				$\begin{bmatrix} -\text{high} \\ +\text{low} \\ +\text{back} \\ +\text{seg} \end{bmatrix}$	[−seg]	*Vowel projection*

More complex phonotactic relations: vowel harmony

Demo: Shona vowel harmony

*[+high][−high,−low] : noninitial [e] w/out harmony trigger prohibited

*[^−high,−low][−high,−low]: [a i u] cannot be followed by [e o]

*[−high,−back][+high,−low,−back]: *eo

..etc

Summary

- Maximum entropy approaches provide a convenient, principled way to model constraint-based grammars
- Naturally produce probabilistic output, which can be compared to probabilistic linguistic judgments, frequencies, etc.
- However, don't displace the need for phonological analysis, as indicated by the failure of the baseline model to deal with the nonlocal phonotactics of Shona.