A Guide to Analysis in MaxEnt Optimality Theory

Chapters 1 and 2

partial book draft, February 2025

Bruce Hayes and Claire Moore-Cantwell

Department of Linguistics

UCLA

Reader comments would be most welcome, including suggested citations. Please send them to bhayes@humnet.ucla.edu and/or clairemoorecantwell@gmail.com.

Feel free to use this material in teaching.

We plan to offer updates on LingBuzz as new chapters are completed, eventually producing a published hard-copy book.

Contents

1. Chapter 1: Purpose and orientation

1.1 Rationale for this book

MaxEnt grammars (Smolensky 1986, Goldwater and Johnson 2003, et seq.) are a formal apparatus for constraint-based linguistics. MaxEnt is is an outgrowth of Optimality Theory (Prince and Smolensky 1993), but works with probabilities, making it suitable for dealing with gradient phenomena. These include free variation (multiple outputs from a single input), gradient well-formedness (e.g., in syntactic judgments or phonotactics), and the matchup of native speaker judgments to statistical patterns found in their language. MaxEnt grammars can also be used in providing concrete interpretations for linguistic experiments. Increasingly, linguists having been using MaxEnt for all of these purposes.

We will start by explaining why we have written this book. On various occasions, we have found ourselves conversing with linguists who have gathered interesting data that might have important implications for theory, yet are not in a position to construct a rigorous analysis of their data because they are quantitative in character. Our intent is to help such linguists overcome the barriers to using MaxEnt grammars for this purpose. MaxEnt offers both quantitatively precise analysis as well as statistical testing, which permits each constraint in the analysis to be checked for whether it is making a meaningful contribution.

We are phonologists and have primarily used phonological examples here. But, we think linguists working in other branches of the field might also find the approach here helpful to their research. In Chapter 8 we survey examples from other areas of linguistics: phonetics, metrics, morphology and syntax, in which researchers have pursued a MaxEnt approach or something like it.

1.2 MaxEnt as tool vs. theory

We here consider MaxEnt from two perspectives. The first is the essentially practical one of our needing analytical tools to help us come to grips with quantitative linguistic data. We believe that MaxEnt can be employed by linguists to obtain closer, deeper understanding of their data, and indeed our confidence that this is so is increased by our experience in guiding linguists who possess interesting data to use MaxEnt to understand their data better. In short, we think it is a great tool.

However, many linguists who have worked in MaxEnt have a more ambitious claim, namely that MaxEnt is a (partial) theory of how language works. Like other serious linguistic theories, MaxEnt has testable claims at stake, which can be extracted from the theory's basic principles through reasoning and testing against language-particular phenomena and typology. We share this interest, and discuss the testable predictions of MaxEnt as a theory of language — not just for phonology, but throughout linguistics.

This distinction between tool and theory in part guides our exposition, which begins as a kind of how-to, and only gradually (§9, but prefigured in §2.8), moves on to the theory's predictions and their assessment.

1.3 Why is it called "MaxEnt"?

One stumbling blockFor this theory is its name, "Maximum Entropy," which in the context of linguistics may come across as alien or even pretentious. This is not a problem we are in a position to fix, but in brief: "Maximum Entropy" in the context of linguistics designates the mathematical apparatus that was attached to Optimality Theory by Goldwater and Johnson (2003) to render it probabilistic. In current linguistics usage, "Maximum Entropy" or "MaxEnt" are often used to refer to the whole package; i.e. the OT architecture coupled with the MaxEnt math. The intellectual connection to actual entropy (as in physics or information theory) is, in practice, rather remote and perhaps best ignored at this stage. For further discussion of the term see §4.8.

1.4 Content

Chapter 2 begins the tutorial material, starting with a review of classical Optimality Theory, the framework from which MaxEnt derives. A key aspect of the tutorial is that MaxEnt analyses are easily carried out on an ordinary spreadsheet. We will suggest that this turns a generic desktop application, which most people use anyway, into a powerful everyday tool for linguistic analysis. We work with an ascending series of concocted languages based on a real one, Khalkha Mongolian.

Chapter 3 extends the tutorial with an example from Tagalog, and illustrates the analysis of lexical frequency matching.

Chapter 4 gives the substance of the MaxEnt framework, including the math. The focus is on intuitive understanding of what the math is doing, with a focus on the characteristic behaviors of MaxEnt grammars and how they may be said to embody sensible inductive reasoning. We also cover the method by which the best-fit weights for a MaxEnt analysis are obtained.

Chapter 5 shows how MaxEnt analysis can be provided with the statistical rigor that is common in the quantitative sciences. We emphasize both visual scrutiny of the analysis and statistical tests that can tell us when the constraints used in an analysis are effective beyond what might simply be due to chance.

Chapter 5 resumes the tutorial material with datasets, here focusing on "static" patterns. Here, MaxEnt is used to define inventories of well-formed entities, such as legal phonological words, and to capture nuances of well-formedness among these entities. The methods used earlier; spreadsheet calculations and statistical assessment, are also applicable here. Our data example comes from Turkish vowel sequencing.

Chapter 6: Since Wilson (2006), MaxEnt has been used to test hypotheses about the language faculty (UG) through the use of **bias modeling**: the MaxEnt grammar is trained not solely from the data, but with a prior expressing a hypothesis about UG. Ideally, such modeling can explain cases where native intuition mismatches the data encountered during learning. Our data example comes from Hungarian vowel harmony.

Chapter 8 gives a survey of MaxEnt research in domains beyond what is covered here (alternations and phonotactics). Within phonology, we discuss loanword adapation, lexical strata, and underlying representation inference. Beyond phonology, we mention work in metrics, phonetics, morphology, and syntax.

Chapter 9: In this chapter, we take up MaxEnt in its role not of analytical tool but of theory: it cannot analyze just anything, but makes its own predictions — often at a rather abstract level — about what can occur in the world. Notably, it naturally derives certain functions that express patterns often seen in linguistic data. We discuss some of these functions and the degree to which they have real-world counterparts. We also cover some critiques of MaxEnt that have been made.

1.5 Our MaxEnt spreadsheets and their locations (Supplementary Materials)

We advocate the use of spreadsheets for MaxEnt calculations, favoring them for their transparency and ease of use. We have placed a variety of spreadsheets discussed in the text on the internet, from which it can be downloaded. Here are the locations from which the spreadsheets are currently available:

https://osf.io/r34bu/ (Open Science Foundation)
https://brucehayes.org/GuideToMaxEnt/

In the text we refer to the online files as the Supplementary Materials.

1.6 Policy on mathematical presentation

MaxEnt involves a certain amount of math. It can be used by a linguist as a tool without understanding any of it, but is more satisfying and reliable when one does understand the math. We address a broad linguistic audience here and seek to be helpful to readers with a broad range of mathematical background. The text itself is directed to an imagined reader who studied algebra in high school, as well as (perhaps) a bit of calculus, and when reminded of various theorems and formulas taught in these courses will be more or less able to recall them to mind. Since we are writing with this imaginary person in mind, the exposition will be a bit usual: we try to show and justify all the steps. This is impossibly verbose by the standards of a technical publication, but far more explicit than is given in (for example) popular science magazines. For readers whose mathematical background is more extensive, we beg patience with our verbosity. We have also tried to label those mathematical sections that can be skimmed without much danger.

2. Chapter 2: The analysis of variation in outputs

We begin with material likely to be familiar to many readers — classical Optimality Theory (OT; Prince and Smolensky 1993) — then develop the exposition by stages to cover, in the end, a nontrivial MaxEnt analysis. For pedagogical purposes we work with a sequence of imaginary languages, to be called *K1* through *K5*. These languages incorporate successively more complex data patterns, for which we employ first classical OT, then OT with free rankings (Kiparsky, 1993; Anttila, 1997), and finally MaxEnt itself. This main thread of the presentation occupies §2.1-§2.7.

The K1-K5 languages form successively better approximations to a real language, namely Khalkha Mongolian, spoken largely in Mongolia. For real Khalkha, our reference source is Fuller (2023), which includes substantial corpus data and its own detailed MaxEnt analysis. Our K1-K5 are intended to build up gradually, ultimately approximating the full complexity seen in Fuller's research. In §2.7.2 we briefly mention how real Khalkha goes beyond our hypothetical languages.[1]

2.1 Analyzing the K1 language in Classical OT

The key (indeed, only) phenomenon of our first hypothetical language, K1, falls under the category of *lenition*, whereby stops and other obstruent sounds take on weaker degrees of closure. Lenition is a common process across languages (Lavoie 2001, Gurevich 2011, Kirchner 2013); some classical instances include the lenition of Spanish /bdg/ to [βðɣ] (Harris 1969), or of English /td/ to the tap [ɾ] (Hayes 2008:31-32). In our hypothetical K1 language, we will suppose an underlying phoneme /p/ that is in non-phrase-initial position is normally lenited to the homorganic glide [w], as in (1a) below. The pattern involves two complications. First, lenition is blocked whenever the /p/ is preceded by a nasal, as in (1b). Typologically, it is common for nasals (which, articulatorily, are stops) to disfavor nonstop articulations immediately after them; thus Spanish, with lenition of /b/ to [β] in many contexts, does not lenite /b/ after [m]; cf. [bomba] 'bomb or pump'. Second, the lenition process is confined to function words; when a /p/ appears in a content word, it never alternates; see (1c).

(1) *The patterning of /p/ lenition in K1*

a. Default realization in non-phrase-initial position: [w]

/mʊntəg **p**oɮ-ən/ → [mʊntəg **w**oɮ-ən]
great become-NPST

'will be great'

[1] At a personal level, we are pleased to have assisted, in a small way, with the creation of Fuller (2023) by suggesting an initial path for the MaxEnt analysis that Fuller built on. In writing this book, we hope to be able to similarly assist other researchers who are in command of interesting patterns of gradient data and seek to create a rigorous generative analysis for it.

b. Retention of [p] in content words

/ɔrʊɮ-ə**x** **p**ɔɮəm-t͡ʃʰɔi / → [ɔrʊɮə**x** **p**ɔɮəmt͡ʃʰɔi]
involve-Fut opportunity-have

'will have the opportunity to involve'

c. Retention of [p] after nasals

/ʃit-t͡ʃʰə**x**-sən **p**ai-sən/ → [ʃitt͡ʃʰəxsə**n** **p**aisən]
decide-Perf-Pst be-Pst

'had decided'

The pattern just given is the sort to which classical Optimality Theory (OT) has often been applied, and it reveals in a particularly clear way the basic ideas behind OT in action. We give here only the briefest review of classical OT. Readers who want better background in this theory are advised to consult the work cited at the end of this chapter, under Further Reading. Much of what MaxEnt has to say reflects its status as an outgrowth of OT, and indeed many of the controversial issues for MaxEnt are controversial for OT as well.

In OT, the problem of phonological derivation is approached as searching for a particular *candidate output* that best satisfies a ranked hierarchy of phonological constraints. This *optimal* candidate is taken to be the (unique) output of the grammar. The theory assumes a function GEN that provides a full set of candidates — in the limit, the infinite set of all possible strings. The function EVAL takes in a phonological input, the candidates of GEN, along with the ranked constraint set, and outputs a winner. Candidate selection works as follows: if the highest ranked constraint C that treats candidates $Cand_1$ and $Cand_2$ differently is violated more times by $Cand_2$ than $Cand_1$, then $Cand_2$ cannot be the winner. Similar comparisons, applied repeatedly under a suitable constraint set, will normally return a unique winning candidate.

We illustrate with K1. We will make use of four constraints, as follows. First, in order to account for the lenition in the first place, we will use the constraint *p:

(2) *p

Assign a violation to every non-phrase-initial [p].

This is can be taken as one member of a larger family that penalizes segments that impose high articulatory effort, such as stops; see for instance Kirchner (1993)'s constraint LAZY.

If *p were the only constraint in the grammar, then of course all underlying /p/ would be converted, often wrongly, to some other sound. But in K1 the process is "held back" in two particular contexts, namely in content words (1b) and after /m/ (1c).

We analyze the resistance to alternation in content words using the well-known family of *positional faithfulness* constraints (Beckman, 1997; for constraints distinguishing content from function words see Casali, 1997, Alderete, 2003). Faithfulness constraints penalize candidates that change something in the input; and the standard taxonomy of Faithfulness constraints, McCarthy and Prince (1995), is a systematic enumeration of all the possible (minimal) changes

that could affect a phonological representation. Here, for simplicity, we assume that [p] and [w] differ solely in the feature [sonorant],[2] and our crucial Faithfulness constraint will be called IDENT(sonorant)$_{content}$.

(3) IDENT(sonorant)$_{content}$

Output correspondents of an input [αsonorant] segment in a content word word are also [αsonorant].

We can now begin to see the rationale for constraint ranking: *p alone will cause /p/ to be lenited everywhere except initially, but if *p is outranked by IDENT(sonorant)$_{content}$, lenition will not occur in content words, but will be confined to function words.

A comparable analysis can cover the other case of lenition blockage — after /m/ — discussed above. We formalize this with a constraint of the widely-used AGREE family (Lombardi 1996, 1999; Bakovic 2000) which penalizes segment sequences that differ in a particular feature. Here, the constraint needed is AGREE(continuant), penalizing the bad candidate *[mw] for underlying /mp/; this is because [m] is [−continuant] and [w] is [+continuant].

(4) AGREE(continuant)

Adjacent segments should have the same value for [continuant]

It is plain that AGREE(continuant) must dominate *p; or, to use the normal notation, AGREE(continuant) >> *p.

We now assemble the ingredients for an analysis: three constraints, plus schematic representative forms that, if properly derived, lead us to expect that the relevant forms of K1 can be derived in general. We are specifically interested in inputs where (a) /p/ is non-initial, but occurs in a content word; (b) /p/ is non-initial, but follows /m/; (c) /p/ is noninitial and neither of these two conditions are met. We want a form with [w] to be the winning candidate in case (c), but [p] should win in cases (a) and (b).

Aside from these constraints, we will also include in our analysis the constraint IDENT(sonorant) (i.e. the general case, *not* limited to content words). This constraint is needed to ensure that phrase-initial /p/ does not also lenite. We will not model this explicitly here, but include the constraint for completeness, and because it will become important later when variation is introduced. In general, careful analysis in OT includes constraints violated by "unfaithful winners" (McCarthy, 2008, §2.2). It should be clear that *p must dominate IDENT(sonorant); else lenition to [w] would never happen at all.

[2] In the feature set of Hayes's (2008) text, they also differ in quite a few additional features: [voice], [round], [continuant], [consonantal], [approximant], [high], [back], [dorsal], and [tense]. A full analysis would include IDENT constraints for these features, as well as constraints expressing the impossibility of other possible outcomes such as [β] or [ʍ] that could repair a non-initial /p/.

We can now provide candidate tables, often called "tableaux". At the top of each tableau, we list the constraints in an order compatible with the necessary pairwise rankings. Tableau rows give candidates from GEN, and asterisks indicate violations. It is easy for the eye to scan across such a tableau and observe how, going from left to right, the losing candidates duly drop out of the competition, leaving the winner (indicated with the traditional "pointy finger") in place at the end.[3]

(5) *Tableaux for K1*

a. Simple lenition in a function word

	/mʊntəg **p**oɮ-ən/ great become-NPST	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)
	mʊntəg **p**oɮən			*!	
☞	mʊntəg **w**oɮən				*

b. Blockage of lenition in a content word

	/ərʊɮ-əx **p**əɮəm-t͡ʃʰɔi / involve-FUT opportunity-have	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)
☞	ɔrʊɮəx **p**əɮəmt͡ʃʰɔi			*	
	ɔrʊɮəx **w**əɮəmt͡ʃʰɔi	*!			*

c. Blockage of lenition after nasals

	/ʃit-t͡ʃʰəx-sə**n** pai-sən/ decide-PERF-PST be-PST	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)
☞	ʃitt͡ʃʰəxsə**n** **p**aisən			*	
	ʃitt͡ʃʰəxsə**n** **w**aisən		*!		*

We note that the rankings that were actually needed were really just *p >> IDENT(sonorant) (so lenition will happen); AGREE(continuant) >> *p (so lenition won't happen after nasals), and IDENT(sonorant)$_{content}$ >> *p (so lenition won't happen in content words). The constraints AGREE(continuant) and IDENT(sonorant)$_{content}$ never conflict with each other (both militate *against* lenition) and so there is no basis for assigning them a ranking (the order we pick for appearance on the page is arbitrary). The inessential ranking is shown with the dotted line in the tableaux of (5).

Our analysis thus far has been carried out using a relatively trivial GEN, with only two candidates for each input. A more valid analysis of K1, or any language, would select the right

[3] We follow standard practice in other aspects of our tableaux: an exclamation point marks the particular asterisk (sometimes not the first in its cell) that kills off a candidate. Gray shading indicates cells which are irrelevant to evaluation, because the winning candidate has already been selected somewhere to the left.

candidate out of an infinite GEN, and would require many more constraints and rankings. For example, in (5a.) the candidate [mʊntəg oɮən], in which the offending [p] has been deleted, rather than just lenited to [w], could be ruled out by ranking a MAX constraint (McCarthy and Prince, 1995) over IDENT(son). The somewhat more trivial candidate [mʊntəg **ə**poɮən], with an epenthetic schwa that does not help with the violation of *p, would be ruled out with a DEP constraint, as would even more bizarre candidates, like [**ʊ**mʊntəg poɮən] or [mʊntəg poɮən**t**]. In actual analytical practice scholars tend to present a GEN that includes anything that seems reasonable and bears on the problem at hand, but they must be on guard that they haven't missed something. Combinatorics, and software, can be helpful here (Karttunen 2006, Riggle 2009). We discuss some issues of GEN specific to MaxEnt in §6.2 below.

OT has been extremely influential in phonological theory, adopted by many who had previously made use of the rule-based framework of *SPE* (Chomsky and Halle 1968). Before we go on, we mention two reasons why this is so.

First, OT accomplishes what many feel to be a scientifically sensible goal, namely stripping down the analysis of phonological systems to their minimal ingredients. Here, these are the Markedness principles that militate against particular surface forms and the Faithfulness constraints that atomize the set of ways in which surface forms can differ from their underlying forms. In contrast, rule-based theory deploys, in essence, *bundles* of Markedness-cum-Faithfulness: a rule like A → B / C___D combines a tacit "phonological problem" (a.k.a. Markedness constraint) that can be expressed as *CAD, together with the licensing of the change from A to B to solve this problem (Prince and Smolensky 1993:4). That such generality is worth achieving becomes evident when we note that the same Markedness constraint gives rise to different Faithfulness-violating remedies, both across languages (Kiparsky 1973) and across contexts within the same language (Kisseberth 1970). Beyond this, a Markedness constraint sometimes can have the effect of *blocking* a change in some languages but *causing* it in another. Thus AGREE(continuant) blocks Lenition in K1 and Spanish; but actively causes fricatives to become stops in languages with "postnasal hardening" (Rosenthall 1989).

A further benefit from OT's separation of Markedness problems from Faithfulness remedies is that we can make language-specific phonological analysis more responsible to typology: when things go well in OT (as they often do), we find that a very detailed language-specific analysis can be reduced to a language-specific ranking of principles each of which has broad cross-linguistic support. In the present case, we find that three principles all (with suitable library research) would be found to have extensive precedents beyond K1 (or more precisely, the real-life Khalkha on which K1 is based.) Stops lenite non-initially in a great number of languages, homorganic nasals frequently impede lenition, and content words often resist alternation more than function words do. Thus our analysis of K1 can be argued to have been assembled from "wholesome," typologically-sensible ingredients. At the very least, OT forces the analyst to declare which constraints in the analysis have broad cross-linguistic support and which ones are parochial; as we will see in Chapter 7, this distinction can have detectible consequences.

2.2 K2 and free variation

In our next hypothetical language K2, pretty much everything is the same, except that the process of /p/ lenition is optional; hence while it is possible for a native speaker of K2 to say

[mʊntəg **w**oɮən] for item (1a), it is also possible to say [mʊntəg **p**oɮən]; and similarly for all similar items. Since both are grammatical in the language, but no meaning difference arises, the phenomenon is commonly referred to as *free variation*.

Free variation is perhaps more common in language than some linguists (trained on problem sets) may be aware. The field of **variationist linguistics**[4] (see Labov, 1969, 1972; Wolfram, 1993; Guy, 1993; Walker, 2010) has long studied free variation processes and achieved careful documentation of them in a variety of languages. Our own impression is that even in work with individual language consultants, free variation will often manifest itself provided the researcher is alert to the possibility — it often pays to ask the consultant "is also possible to say it this way?". Typically, carefully analysed free variation is not a sea of chaos, but is phonologically structured (Cedergren & Sankoff, 1974; Sankoff, 1988), in ways that we will suggest are well suited to being addressed with MaxEnt grammars.

For the simplest cases of free variation, there is an immediately obvious possible approach, which occurred to theorists (notably Kiparsky 1993, 2006ab; Anttila 1995, 1997) almost as soon as OT was invented: instead of imposing a full ranking of the constraints in the grammar, we let certain constraint pairs (and perhaps triplets, etc.) be *ranked freely*. In the case of K2, we would posit free ranking for the constraints *p and IDENT(sonorant), as shown in the pair of tableaux in (6) below. Note both tableaux retain the former free ranking of $\text{IDENT(sonorant)}_{\text{content}}$ and AGREE(continuant) — this is "trivial" free ranking, in that it has no effect on the outcome. The essential free ranking is that of *p and IDENT(sonorant), since when these are ranked differently, different candidates win.

(6) *A tableau pair for a case in which two constraints are freely ranked*

	/mʊntəg **p**oɮ-ən/ great become-NPST	$\text{IDENT(son)}_{\text{content}}$	AGREE(cont)	*p	IDENT(son)
	mʊntəg **p**oɮən			*!	
☞	mʊntəg **w**oɮən				*

PLUS

	/mʊntəg **p**oɮ-ən/ great become-NPST	$\text{IDENT(son)}_{\text{content}}$	AGREE(cont)	IDENT(son)	*p
☞	mʊntəg **p**oɮən				*
	mʊntəg **w**oɮən			*!	

This use of multiple tableaux is verbose, and normally we just write one, with a dotted vertical line separating the freely ranked constraints, just as it does in (6) for $\text{IDENT(sonorant)}_{\text{content}}$ and AGREE(continuant).

[4] Also called **sociolinguistics**, especially in earlier work, though that term is now used more broadly. See Walker (2010) ch.1 for discussion.

(7) *A single tableau with free ranking that predicts multiple outputs*

	/mʊntəg **p**oɮ-ən/ great become-NPST	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)
☞	mʊntəg **p**oɮən			*!	
☞	mʊntəg **w**oɮən				*

In (7), we have just one pair of freely ranked constraints affecting the outcome, but the theory permits in principle any number of pairs (or triplets, quadruplets, etc).

2.3 The interpretation of free ranking as probability with partially ranked constraints

The grammar illustrated in (7) is a rough-and ready account of free variation: it clearly tells us that two outcomes are possible. We can think of this grammar as one that simply accepts both [mʊntəg **p**oɮən] and [mʊntəg **w**oɮən]. Such a grammar would not make any predictions about which outcome was more common, and under what circumstances. However, in the work of Anttila (1997) and Kiparsky (1993, 2006ab), grammars like the one illustrated in (7) are used to predict precise probability distributions over candidate outputs.

This body of work proposes and refines an important and still-advocated theory of variation in OT. It has various names, (Anttila, 1997; Kiparsky, 2005; Riggle, 2010) but we will call it the **Theory of Partially Ranked Constraints**, following Anttila. To explain how the theory works, we use the example of the tableau in (7). Going across the constraints in this tableau from left to right, we let each group of constraints delimited by solid lines be called a *constraint stratum*. Thus for (7) the constraint strata are as in (8).

(8) *Constraint strata for tableau* (7)

a. *First stratum*

IDENT(son)$_{content}$, AGREE(cont)

b. *Second stratum*

*p, IDENT(son)

Within each stratum (no matter how many constraints are in it), ranking is assumed to be free. Further, it is assumed that any constraint in a higher stratum must outrank all constraints of a lower stratum. In order to predict probabilities over outputs, we first need to count the rankings compatible with these strata. Here, each stratum with two constraints in it gives rise to two possible rankings of those constraints; and because we have two strata each with two freely ranked constraints in them, the total number of rankings is $2 \times 2 = 4$. To be specific, the four rankings are as given in (9):

(9) *Rankings compatible with* (8)

a. IDENT(son)$_{content}$ >> AGREE(cont) >> *p >> IDENT(son)
b. IDENT(son)$_{content}$ >> AGREE(cont) >> IDENT(son) >> *p
c. AGREE(cont) >> IDENT(son)$_{content}$ >> *p >> IDENT(son)
d. AGREE(cont) >> IDENT(son)$_{content}$ >> IDENT(son) >> *p

In the general case, a stratum with *n* constraints gives rise to *n! (n* factorial) possible rankings. We can then calculate n_i! for each freely ranked stratum *i*, then multiply out all of these terms to find the ranking count.

Now that we can count rankings, we can say that the *probability* assigned to any candidate under the theory of partially ranked constraints is simply the number of rankings for which that candidate is the winner, divided by the total number of rankings. The actual computation of this probability when the constraint set is intricate may be difficult, but the underlying concept seems straightforward. In the case of (7), there are four possible rankings (2! times 2!), of which two derive [mʊntəg **p**oɮən] and two derive [mʊntəg **w**oɮən].[5] Hence the prediction is that the lenited form [mʊntəg **w**oɮən] will surface exactly half the time, a probability of 0.5. Note that no ranking produces lenition in content words, or after nasals, so surface forms like *[ʃɪtt͡ʃʰəxsə**n** **w**aisən] or *[ɔrʊɮəx **w**əɮəmt͡ʃtʰɔi] are predicted to have a probability of zero.

What we have just established is sufficiently important that it is worth describing it in more general terms: unlike OT, a grammar with partially ranked constraints is a **probabilistic grammar**. Instead of just selecting one or more candidates from GEN as grammatical outputs for the given input, it assigns probability values (often zero) to the members of GEN. It is probabilistic grammars that we will concern ourselves with for the rest of this book, as we seek to find analyses for K2 and other, more complex cases.

Grammars using partially ranked constraints are, in a sense, the minimal extension of classical OT that can be used to create probabilistic grammars. They have been adopted for the analysis of a variety of systems with free variation (see references above, as well as Pater 2000, and Zuraw 2002).

We admire this approach for its simplicity, but nevertheless we think it is wrong. It is necessary that the theory be supported not just by isolated analytical successes, which could arise by accident, but by *consistent* success. Cases where the theory fails rather badly are given in Zuraw and Hayes (2017).

2.4 K3: modeling arbitrary probabilities

Idealized data like that in K2 are very common, often resulting from qualitative examination of speakers' intuitions. However, when researchers turn to quantitative examination of large

[5] Recall that the two constraints of the first stratum, IDENT(son)$_{content}$ AGREE(cont), do not conflict, so that their ranking does not matter.

datasets (texts, corpora, dictionaries, or datasets collected from speakers), it is common to find that variants are *not equally likely*. Such cases are the simplest that justify the move beyond partially ranked constraints toward MaxEnt.

When the variants are not equally likely, it will not suffice to say "both outcomes are possible" —this does no justice to the difference between, say, 50%/50% and 75%/25%, or even 98%/2%. In practice, percentages observed in speech and text cover a wide range, and have in various cases been shown to be stable and replicable both in corpora and in speakers' behavior in experiments (see Zuraw 2001, 2010; Ernestus and Baayen 2003; Albright and Hayes 2003; Hayes and Londe 2006; Hayes et al. 2009; Becker 2009; Becker et al. 2011; Becker and Gouskova 2016; Moore-Cantwell, 2020; Garcia, 2019) Because these percentages appear not just in corpora but seemingly in speakers' productive phonology, it is necessary to make use of a model of phonological grammar which can represent these patterns effectively.

To explore this possibility, we introduce our third hypothetical language, K3, which will involve us with quantitative data. To give a sense of realism, we look ahead to the discussion (§2.7.2) of real Khalkha, as studied by Fuller (2023). Fuller obtained her data by carefully transcribing the phonetic realization of every instance of phonemic /p/ in four TEDx talks (https://tedxulaanbaatar.com/), totaling 53 minutes of speech, given by native speakers of Khalkha in Ulaanbaatar, the capital city of Mongolia. From these data, Fuller collected 1334 instances of /p/, sufficient to shed light the quantitative patterning of /p/ lenition in relatively formal Khalkha speech. Our synthetic data for K3 follow the same trends that Fuller observed, but have been constructed to be a maximally clear pedagogical example.

In K3, when a function word is preceded by a non-nasal sound, as in /mʊntəg **p**oɣ-ən/ above, our constructed data include 300 instances of [p] and 110 instances of [w]. Examples are given in (10). Crucially, surface [p], and surface [w] are not equally likely (as they are in the tableau in (7) above, with partially-ranked constraints, would predict); in fact, the probability of weakened [w] is 110/(110+300) = 0.27. The remaining data are similar to K1 and K2, i.e. lenition never occurs in content words or after nasals.

(10) *Synthetic data for K3*

UR type	*Example*	*SR*	*count*
...[−nasal] [$_{\text{function}}$ p...	/mʊntəg **p**oɣ-ən/ great become-NPST	p	300 (73%)
		w	110 (27%)
... [$_{\text{content}}$ p...	/ərʊɣ-əx **p**ɔɣəm-t͡ʃʰɔi / involve-FUT opportunity-have	p	720 (100%)
		w	0
...[+nasal] [p...	/ʃit-t͡ʃʰəx-sə**n** pai-sən/ decide-PERF-PST be-PST	p	212 (100%)
		w	0

In order to model the data of K3, we seek a probabilistic grammar which can predict the exact lenition rates that we see in the data: 0% in content words and after nasals, and 27% in function words. This will consist of a MaxEnt grammar, so we first need to take a fairly substantial detour to explain what a MaxEnt grammar is.

2.5 MaxEnt: an introduction

Maximum Entropy Grammar is a variant of Optimality Theory, designed for the purpose of formulating probabilistic grammars. The key difference is that instead of using constraint ranking, MaxEnt assigns a numerical value to each constraint, its **weight**, which intuitively reflects its strength. We reemphasize here that MaxEnt is very much like OT (§4.1.1): it evaluates candidates in parallel, uses a potentially infinite GEN, and atomizes phonological analyses into simple, independent constraints; thus solving the conspiracy problem and linking language-specific analyses to phonological typology.

2.5.1 Preview: Classical Harmonic Grammar

To present MaxEnt clearly, we will begin with a simpler theory which will serve as a kind of way-station: **Classical Harmonic Grammar**, a theory originating with Legendre et al. (1990). In Classical Harmonic Grammar, the constraint weights and violations are used to calculate a **Harmony** score for each candidate, which (despite the name) acts as a kind of penalty. The output of the grammar in this theory is the least-penalized member of GEN for a given input. Note even though Classical Harmonic Grammar uses numbers, it is *not* a probabilistic theory: it produces one single winner for each candidate competition, just as in Optimality Theory.

The Harmony score is calculated as follows:

(11) *Calculating the Harmony of a candidate*

a. For each constraint, multiply the number of violations the candidate incurs by the constraint weight.
b. Sum the result obtained in step (a) across all constraints.

The same computation is expressed as a formula in (12):

(12) *Formula: Harmony for candidate* x

$$\mathcal{H}(x) = \sum_{i \in Constraints} w_i * v_i(x)$$

where v_i(x) is the number of violations incurred by that candidate for constraint *i*.

We illustrate with an example. Since the theory allows only one single winner per input, we we return to the data of K1 ((1) above), and derive the pattern in Classical Harmonic Grammar. The crucial tableau is (13), which is a Harmonic Grammar version of (5). For this tableau, we have hand-selected the weights for each constraint.

To give an example of Harmony calculation, candidate [mʊntəg **p**oɮən] violates *p just once. The weight of *p is 2, hence the Harmony contribution from this constraint is 2. Since [mʊntəg **p**oɮən] *only* violates *p, the total Harmony score is therefore 2. Candidate [ʃitt͡ʃʰəxsə**n** **w**aisən] incurs one violation of AGREE(cont), which has a weight of 3, and one violation of IDENT(son), with a weight of 1. So its Harmony is calculated as 1 × 3 + 1 × 1 = 4.

Once we have calculated Harmony, it is straightforward to determine the unique winner for each input, namely the candidate with the lowest Harmony penalty.

(13) *Classical Harmonic Grammar tableau for K1*

	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)	
weights:	3	3	2	1	Harmony
/mʊntəg **p**oɮ-ən/ great become-NPST					
mʊntəg **p**oɮən			1		2
☞ mʊntəg **w**oɮən				1	1

/ɔrʊɮ-əx **p**ɔɮəm-t͡ʃʰɔi / involve-FUT opportunity-have					
☞ ɔrʊɮəx **p**ɔɮəmt͡ʃʰɔi			1		2
ɔrʊɮəx **w**ɔɮəmt͡ʃʰɔi	1			1	4

/ʃit-t͡ʃʰəx-sə**n** pai-sən/ decide-PERF-PST be-PST					
☞ ʃitt͡ʃʰəxsə**n** **p**aisən			1		2
ʃitt͡ʃʰəxsə**n** **w**aisən		1		1	4

In (13), each constraint is only violated once. To illustrate the multiplication of violation counts by weights, we consider another example from Fuller (2023), which contains multiple p's.

(14) *Multiple lenition sites in a single input*

odoo khümüüs oroi	***untdag***	***polson***	***pai-na***
	/ʊntʰ-təg	pɔɮ-sən	pai-n/
now people late	sleep-HAB	become-PST	be-NPST

'Nowadays, people sleep late.'

The harmony of neither, just one, or both /p/'s leniting is shown in (15):

(15) *Calculating Harmony in Classical Harmonic Grammar with multiple constraint violations*

	IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT(son)	
weights:	3	3	2	1	Harmony
/ʊnt^h-təg **p**ɔɮ-sən **p**ai-n/					
ʊnt^{h}təg **p**ɔɮsən **p**ain			2		4
☞ ʊnt^{h}təg **w**ɔɮsən **p**ain			1	1	3
ʊnt^{h}təg **p**ɔɮsən **w**ain		1	1	1	6
ʊnt^{h}təg **w**ɔɮsən **w**ain		1		2	5

As can be seen, for candidate [ʊnt^{h}təg **p**ɔɮsən **p**ain], the constraint *p contributes 4 units of Harmony, the product of two violations and its weight of 2. The winner here is actually [ʊnt^{h}təg **w**ɔɮsən **p**ain], with lenition on the first, but not the second, /p/. This one violates *p only once, and IDENT(son), which has a lower weight, once. This makes its harmony just 3, lower than the 4 of the fully faithful candidate. Either candidate where the second p is lenited (the last two) also violates AGREE-Cont, with its higher weight of 3.

2.5.2 Shifting to MaxEnt

The discussion of Classical Harmonic Grammar just given provided the mechanisms for converting weights and violation patterns into Harmony scores. For MaxEnt itself, there is one more step, namely to convert Harmony scores into probabilities. MaxEnt is a probabilistic version of Classical Harmonic Grammar, just as the theory of partially ranked constraints is a probabilistic version of OT. The math we will cover here will have two essential properties. First, it will assign lower probability to candidates that have higher Harmony penalties. Second, the probability assigned to the full set of candidates for a given input will sum to one; this is what it means to be a probability distribution.

In (16), we give the MaxEnt transformation from Harmonies to probability in the form of a recipe.

(16) *Calculating probability from harmony values in MaxEnt*

a. Calculate the **Harmony** of each candidate, as in (12).

b. For each Harmony value, first make it negative, and then exponentiate it using Euler's number *e*.[6] We'll call this value **eH**:

$eH = e^{-H}$

[6] Euler's number *e*, useful in mathematics for many reasons, is about 2.718. MaxEnt would work perfectly well with some other base, such as 10, but we follow standard practice here.

c. For each input, sum up the eH over all the candidates for that input. Call this **Z**.

$$Z = \sum_{i \in candidates} eH_i$$

d. For each candidate, divide its eH by Z. This is its predicted **probability**.

$$p(i) = \frac{eH_i}{Z}$$

To illustrate these mechanisms, we return to K3, the language in which [mʊntəg **p**oɮən] occurs 73% of the time, and [mʊntəg **w**oɮən] 27%. We seek to formulate a MaxEnt grammar that matches this percentage, and also derives the invariant outcome for the remaining two inputs.

Formally, a MaxEnt grammar consists of a set of constraints, each paired with a weight. Our proposed grammar is given in (17).

(17) *A MaxEnt grammar for K3*

Constraint	*Weight*
*p	3
IDENT(son)	4
IDENT(son)$_{content}$	20
AGREE(cont)	20

The weights in (17) were chosen by hand. While it is possible to find a good fit to the data by choosing weights by hand for a very small dataset like this one, realistic analyses generally require the weights to be fit by machine, a topic taken up in §2.6.

For now we will simply accept the weights of (17) and proceed with the computation. All the needed values appear in tableau (18) below: the weights of (17) are shown underneath their respective constraint names; and the constraints assign violations in the same way as in previous tableaux. To illustrate the calculations, we derive the probability of candidate [mʊntəg **p**oɮən], which competes with [mʊntəg **w**oɮən] as the output for /mʊntəg **p**oɮən/. First, its Harmony is calculated in the same way as in Classical Harmonic Grammar; it comes out as 3 × 1 = 3. To convert this Harmony value into a probability, we follow the procedures in (16). We calculate eH (16b); which comes out to e^{-3} = 0.050. Next, (16c) tells us to calculate Z by summing eH values for both candidates. We calculate eH for the competitor candidate [mʊntəg **w**oɮən], which comes out to 0.018, then add the two eH values to obtain Z; 0.50 + 0.18 = 0.068. Lastly, we calculate the probability of [mʊntəg **p**oɮən] according to (16d), by dividing its eH by Z. This is 0.050/0.068 = 0.73. The last three columns of tableau (18) illustrate the computations for all candidates.

(18) *Tableau: MaxEnt analysis of K3*

			IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT (son)				
weights:	Freq.	prob.	20	20	3	4				
/mʊntəg **p**oɮ-ən/ great become-NPST							H	eH	Z	p
mʊntəg **p**oɮən	300	0.732			1		3	0.050	0.068	0.73
mʊntəg **w**oɮən	110	0.268				1	4	0.018	0.068	0.27
/ɔrʊɮ-əx **p**əɮəm-t͡ʃʰɔi / involve-FUT opportunity-have										
ɔrʊɮə**x** **p**əɮəmt͡ʃʰɔi	720	1			1		3	0.049	0.050	≈1
ɔrʊɮə**x** **w**əɮəmt͡ʃʰɔi	0	0	1			1	24	3.77* 10^{-11}	0.050	≈0
/ʃit-t͡ʃʰəx-sə**n** pai-sən/ decide-PERF-PST be-PST										
ʃitt͡ʃʰəxsə**n** **p**aisən	212	1			1		3	0.049	0.050	≈1
ʃitt͡ʃʰəxsə**n** **w**aisən	0	0		1		1	24	3.77* 10^{-11}	0.050	≈0

As you can see, we have chosen our weights such that the MaxEnt-predicted probabilities closely match the frequencies of our toy data set, given as raw numbers in the second column of (18) and as proportions in the third. The generation of probabilities like 0.73 is beyond the capacity of Classical OT or Classical Harmonic Grammar, which are nonstochastic theories. Using partially ordered constraints would also be insufficient, since only a probability distribution of 50/50 can be generated. MaxEnt, in contrast, can generate any probability distribution in such cases with a suitable choice of weights.

At this point we have illustrated the core of MaxEnt, the math that converts Harmony ($\mathcal{H}$) to probability. To sum up, we reexpress the MaxEnt recipe of (16) as a single formula, in (19).

(19) *Maxent formula (terse version)*

$$P(x) = \frac{e^{-\mathcal{H}(x)}}{Z} \quad \text{where } Z = \sum_i e^{-\mathcal{H}(cand_i)}$$

where x is a candidate for some input
$P(x)$ is the probability the grammar assigns to x

e^t is e taken to the tth power
$\mathcal{H}(x)$ is the harmony of candidate x
Σ_i denotes the sum over all candidates (*cand*)

In a fuller version, we can include instead of $\mathcal{H}$ the actual calculations from (12) by which Harmony is obtained:

(20) *Maxent formula (verbose version)*

$$p(x) = \frac{e^{-\sum_{i \in Con} w_i * v_i(x)}}{\sum_{y \in candidates} e^{-\sum_{i \in Con} w_i * v_i(y)}}$$

where w_i is weight of the ith constraint
$v_i(y)$ is the number of times candidate y violates the ith constraint
Σ_i denotes the sum over all constraints

2.6 K4: Perturbers as "hidden" phonology in variation

We now move to our next hypothetical language, K4. This toy language is intended to represent a very common situation in patterns of variation. Not only is the probability of the two possible outcomes unequal, but additionally it is *affected by context*; that is, there is at least one independent factor — we will call it a **perturber** — that changes the output probabilities in a systematic way (Labov 1969, Bayley, 2002). In a MaxEnt grammar, this additional factor will usually be treated using what we call a "perturber constraint." Candidates that violate a perturber constraint will have a probability distribution different from what is seen among non-violating candidates. It is a strength of MaxEnt that it permits the effects of perturbers to be identified and analysed.

Our K4 language is set up to illustrate the effect of a perturber. As in K3, lenition does not occur in content words or after nasals. However, in other environments the lenition rate is affected probabilistically by the preceding context: specifically, lenition is more likely after a vowel than after a consonant. For example, in an input like /mʊntə**g p**oɮən/, with preceding consonant, lenition occurs about 17% of the time. But in /tʊt^{h}**ʊ p**ai-ga/, with a preceding vowel, lenition occurs considerably more often, 83% of the time. Hence a preceding vowel is a perturber that increases the probability of lenition. The pattern is summarized in (21).

(21) *Synthetic data for K4*

UR type	Example	SR	*count*
... C [$_{\text{function}}$ p ...	/mʊntəg **p**oɮ-ən/ great become-NPST	p	*290 (83%)*
		w	*60 (17%)*
... V [$_{\text{function}}$ p ...	/tʊt^{h}ʊ **p**ai-ga/ missing be.PROGPRES	p	*10 (17%)*
		w	*50 (83%)*
... [$_{\text{content}}$ p ...	/ɔrʊɮ-əx **p**ɔɮəm-t͡ʃt^{h}ɔi / involve-FUT opportunity-have	p	*720 (100%)*
		w	*0*
...[+nasal] [p ...	/ʃit-t͡ʃ^{h}əx-sə**n** pai-sən/ decide-PERF-PST be-PST	p	*212 (100%)*
		w	*0*

Our K4 language is meant to illustrate a very common situation when researchers examine probabilistic phonological patterns (whether this be lexica, corpora, or experiments): While at first it may appear that all varying forms pattern together (as was true in K3), further inspection reveals non-uniformity: after a vowel, we have predominant lenition (83%), whereas after a consonant lenition is far less usual (17%). If we did not examine the specific rates of lenition in these two contexts, but rather pooled them together, we would have failed to notice the effects of the perturber environment.

To formalize the perturber effect, we adopt a commonplace lenition constraint as our perturber constraint, defined in (22):

(22) POSTVOCALICLENITE

Assign a violation for every sequence of the form [+syllabic][−sonorant].

The idea here is that, as in in many languages, K4 disprefers postvocalic stops, and replaces them with a more sonorous consonant, in this case [w].

For the moment, we stick with the constraint weights we used for K3, but add in POSTVOCALICLENITE, with a weight of 2, chosen by hand. The tableaux in (23) illustrate the predictions of this new grammar. The new input /tʊt^{h}ʊ **p**ai-ga/ is now included, for which the POSTVOCALICLENITE constraint affects the predicted probability distribution.

(23) *Tableaux for K4 (hand-fit weights)*

			IDENT(son)$_{content}$	AGREE(cont)	*p	IDENT (son)	POSTVOCLENITE				
weights:	Freq.	prob.	20	20	3	4	2				

a. Simple leniting environment (no perturber)

/mʊntəg **p**oɮ-ən/ great become-NPST								H	eH	Z	p
mʊntəg **p**oɮən	290	0.829			1			3	0.050	0.068	0.73
mʊntəg **w**oɮən	60	0.171				1		4	0.018	0.068	0.27

b. Leniting environment with postvocalic perturber

/tʊt^{h}ʊ **p**ai-ga/ missing be.PROGPRES											
tʊt^{h}ʊ **p**ai-ga	10	0.167			1		1[7]	5	0.007	0.025	0.27
tʊt^{h}ʊ **w**ai-ga	50	0.833				1		4	0.018	0.025	0.73

c. Blockage in stems

/ɔrʊɮ-əx **p**ɔɮəm-t͡ʃ^{h}ɔi / involve-FUT opportunity-have											
ɔrʊɮə**x** **p**ɔɮəmt͡ʃ^{h}ɔi	720	1			1			3	0.050	0.050	≈1
ɔrʊɮə**x** **w**ɔɮəmt͡ʃ^{h}ɔi	0	0	1			1		24	3.77* 10^{-11}	0.050	≈0

d. Blockage after nasal

/ʃit-t͡ʃ^{h}əx-sə**n** pai-sən/ decide-PERF-PST be-PST											
ʃitt͡ʃ^{h}əxsə**n** **p**aisən	212	1			1			3	0.050	0.050	≈1
ʃitt͡ʃ^{h}əxsə**n** **w**aisən	0	0		1		1		24	3.77* 10^{-11}	0.050	≈0

Our predicted probabilities are reasonably close to the observed ones. (We will see in the next section that fitting the weights by algorithm obtains a closer fit.) The key point for now is that we have successfully incorporated the effect of POSTVOCALICLENITE as a perturber

[7] Note that since POSTVOCALICLENITE is assessed for all segment pairs, there are actually many violations present throughout the tableaux. We record only the violations that distinguish candidates; the outcomes remain the same.

constraint: it lowers the output probability of candidates that violate it (here 0.27 for [tʊtʰʊ **p**aiga] vs. 0.73 for [mʊntəg **p**oɮən]), but does not rule out violating candidates entirely.

2.6.1 More on perturber constraints

It is common for constraints that act as a perturber in one language to appear as a fully determinative factor in another language. For example, Spanish is said to be a language in which post-vocalic lenition (of voiced stops) is entirely obligatory: *navidad* /nabidad/ 'Christmas' *must* be [naβiðað], and cannot realize any of the obstruents as a stop. Although our use of postvocalic lenition as a perturber in Khalkha is invented, we cover in §9.1 several real-language pairs of phenomena where a factor that fully determines the outcome in an obligatory process in one language or languages acts as a perturber in another language. To give one example, in Iraqi Arabic a ban on triple consonant clusters (*CCC) is categorical: when such sequences arise across morpheme boundaries, they are obligatorily repaired (Broselow 1980). But in English, the same constraint acts as a perturber; it merely increases the frequency with which CCC clusters are repaired in surface forms. For example, the /t/ of *tanked* [tæŋkt] is deleted more often than the /t/ of *tact* [tækt] (Labov 1989). In the context of MaxEnt, this pretheoretical observation has a clear theoretical interpretation, namely, that variable and categorical phenomena use the same constraint set. Obligatoriness vs. optionality is simply a consequence of the particular weights that have been assigned.

The pervasive appearance of perturbers in variation is perhaps the best argument for the use of MaxEnt or similar frameworks as an analytical tool: without creating a probabilistic analysis of the data, one is liable to miss important parts of the language system. In Classical Optimality Theory, a nonstochastic theory, there is simply no way to treat perturber phenomena. MaxEnt gives us a way to integrate such phenomena into language-specific analysis and, indeed, into the study of typology.

We suggest that MaxEnt might also be of assistance to the working linguist in trying to make sense of new data: it encourages the analyst to seek out gradient generalizations and test them through explicit formal analysis. If you think in terms of a theory that cannot treat perturbers, it is possible that you will miss the perturbers.

2.6.2 Fitting the weights

Establishing the weights is, of course, part of the MaxEnt analysis, analogous to finding a workable ranking in Optimality Theory. However, weighting is different from ranking in that it is almost impossible to do it accurately by hand. We have found it a useful classroom exercise in teaching beginning MaxEnt to ask our students to do hand-weighting; we find that it is possible but generally difficult, and usually leads to a less accurate grammar than we can obtain by using a machine-implemented algorithm. Following standard usage, we use the term **fitting** to denote the process of finding the best available weights for a given set of data and constraints.

MaxEnt weights can be fit by algorithm using a variety of different software types, which we cover briefly in §2.11. For the MaxEnt beginner, we highly recommend the use of Excel for implementing the MaxEnt math, and the Excel Solver add-on for fitting weights. We recommend this approach mainly for transparency. The cells of the spreadsheet display in full the math that

calculates harmony and converts it to probability. Furthermore, the Excel format makes it easy to implement changes to the analysis and understand their consequences. More experienced MaxEnt practitioners may also find the Excel format useful, as it makes it easy to introduce novel elements to the structure of the theory. We will illustrate many examples of this throughout the later chapters of the book. Lastly, Excel spreadsheets can be placed into online Supplementary Materials of a published paper, so that the calculations become freely inspectable, and modifiable, by anyone with access to Excel.

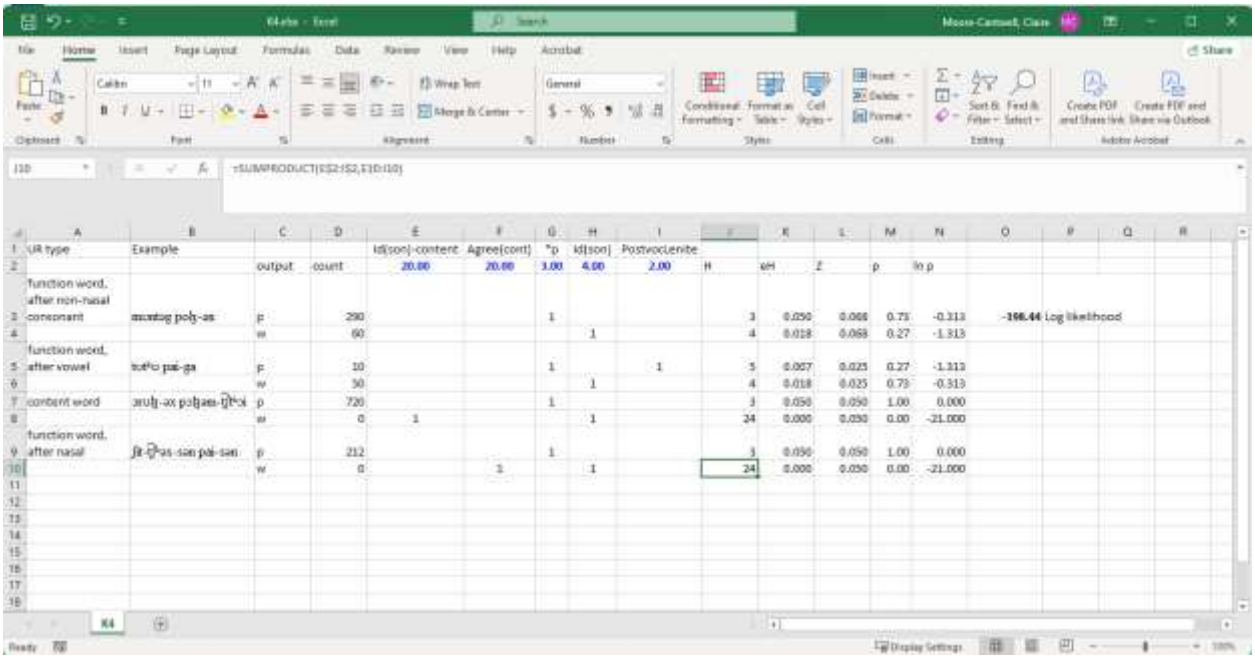
What follows is a "how-to" for implementing a MaxEnt analysis in Excel. We first show how to implement the MaxEnt formula (20) in the format of an Excel spreadsheet (§2.6.3). With this in place, we show how to calculate the weights (§2.6.4).

2.6.3 Implementing the MaxEnt formula on a spreadsheet

The first step is to implement the MaxEnt formula, given above in (20), using the resources of Excel. For this purpose we return to our K4 example. The spreadsheets discussed here may be obtained in the Supplementary Materials for this book, available at the web locations listed in §1.5 above; the particular spreadsheet we will be working with will be "K4.xlsx". This spreadsheet is given in print below, and you may find it helpful not to download it but rather to to type it out yourself as a study aid.

Figure 1 is a screen clip of the actual Excel interface. It is opaque, as one sees the "underlying" values of the cells only when you hover the mouse over them. The content of cell K10 has been selected and thus shows up in the formula bar.

Figure 1: The Excel interface



The table in (24) shows the underlying structure of Figure 1, in particular the formulae that compute the displayed values. We begin at column B, which contains the example inputs of each

type (they appear shortened in (24) relative to the actual spreadsheet). Column C contains the candidate outputs, and column D contains the counts of each candidate output 'observed' in our toy data. Columns E-I contain constraint names in row 1 and weights in row 2. Constraints in columns E-G are not shown, but in the full version of the spreadsheet are as in tableau (23).

(24) *Calculating MaxEnt probabilities in Excel*

	Excel column names									
row #	B	C	D	...	H	I	J	K	L	M
1	Example			...	Id(son)	Postvoc Lenite	Harmony	eH	Z	p
2		output	count		4.00	2.00				
3	g poɮən	p	290				=SUMPRODUCT(E$2:I$2, E3:I3)	=EXP(−J3)	=SUM(K3:K4)	=K3/L3
4		w	60		1		=SUMPRODUCT(E$2:I$2, E4:I4)	=EXP(−J4)		=K4/L3
5	ʊ pai-	p	10			1	=SUMPRODUCT(E$2:I$2, E5:I5)	=EXP(−J5)	=SUM(K5:K6)	=K5/L5
6		w	50		1		=SUMPRODUCT(E$2:I$2, E6:I6)	=EXP(−J6)		=K6/L5
...	...	...	...	...	...	...	...	...	...	...

The next few columns implement the MaxEnt math, closely following the recipe of (16). Column J calculates the Harmony of each candidate, rendering in Excel the formula from (12), $\sum_{i \in Constraints} w_i * v_i(candidate)$. It uses the Excel function SUMPRODUCT(), which performs two steps at once: it multiplies together the individual values for constraint violation counts and weights (columns E-I), then sums the result across all constraints. The expression "`E$2:I$2, E3:I3`" in cell J3 follows standard Excel conventions: `E$2:I$2` means "always the cells in row 2", hence always the row of weights; but "`E3:I3`", since it appears in row 3, means "whatever appears in cells E-I of this row". By using this notation, it is possible to write the function once, in the top cell (here, J3) then copy it down to fill the rest of the column. Excel will automatically update any contextual references, so that the constraint weights will be multiplied by whatever violation counts are present in a given row.

Column K computes the eH value of each candidate *x* (i.e. e^{-H}, from (16b)) by first negating, then exponentiating, the result of Column J. As can be seen, the Excel notation for e^x is EXP(x). As before, this value can be written once and copied down the column.

Column L calculates the normalizing factor Z for each input ((16c)), using the `SUM()` function to total the eH values for all (both) candidates.

Column M carries out the final step, from (16d); i.e. calculating the probability assigned to each candidate by dividing the eH from column K by the Z value from the appropriate cell in column L.

2.6.4 Setting the weights by algorithm

Now that we have expressed the MaxEnt calculations in a spreadsheet, we can turn to our original problem of using software to find the most accurate constraint weights.

To begin, we first have to quantify what we mean by "accurate." We will employ a widely-used metric of accuracy: the **log likelihood** of the data (see, e.g. Jurafsky and Martin (2025, §5.5). Log likelihood is a measure of how likely our observed data are, given our current set of weights. It is defined as the log of the probability of the observed data if the model is assumed to be correct. Log likelihood values are always non-positive (negative or zero), and a higher log likelihood value (i.e., less negative) implies greater accuracy. The search for optimal constraints is now defined as finding the weights that maximize log likelihood. Some important mathematical properties of log likelihood are discussed in §4.7.1 below; for now, it would be safe for the reader to take the discussion as a rote procedure.

We begin in (25) with the formula for log likelihood. We suppose that we are analyzing a complete data set of which d_i represents one single observed datum. For example, the K4 dataset in (21) consist of 1342 data, of which one datum is, for instance, a single token of /mʊnteg poʒ-ən/ → [mʊnteg woʒ-ən]. The formula carries out a computation over the full set of data.

(25) *Formula for log likelihood*

$= \Sigma_i(\ln(P(d_i, \vec{w}))$

where	$P(d_i, \vec{w})$	is the probability assigned by the grammar to the datum d_i using the set of weights $\vec{w}$
	$\ln(x)$	denotes the natural logarithm (base e) of x
	Σ_i	denotes summation across all of the observed data

In Excel, this value can be calculated with a step-by-step procedure, as illustrated in (26). This table shows some additional columns from K4.xlsx, which are used to calculate likelihood. Column M, already seen, gives the predicted probability for each candidate. Column N converts this probability to log likelihood, using the log function (`=LN()`) in Excel. Since many of the data types occur multiple times in the K4 data (see column D), we take account of each datum by multiplying the likelihood values by the frequencies, using the convenient `SUMPRODUCT()` function, which we used earlier to calculate harmonies in column J. The result is given in cell O2.

(26) *Calculating log likelihood in Excel*

	Excel column names									
row #	B	C	D	...	H	I	...	M	N	O
1	Example			...	Id(son)	Postvoc Lenite	...	p	log p	Log likelihood
2		output	count		4.00	2.00				=SUMPRODUCT(D3:D10,M3:M10)
3	ɡ poɮən	p	290					=K3/L3	=LN(M3)	
4		w	60		1			=K4/L3	=LN(M4)	
5	ʊ pai-	p	10			1		=K5/L5	=LN(M5)	
6		w	50		1			=K6/L5	=LN(M6)	
…	…	…	…	…	…	…	…	…		

Note that some candidates are never observed; i.e. their frequency is zero; because of the multiplication, these candidates do not contribute to log likelihood.

When we view K4.xlsx in the normal mode of Excel, we see the values computed by the formulae in (26):

(27) *Same table, with displayed values*

	Excel column names									
row #	B	C	D	...	H	I	...	M	N	O
1	Example			...	Id(son)	Postvoc Lenite	...	p	log p	Log likelihood
2		output	count		4.00	2.00				-198.44
3	ɡ poɮən	p	290					0.73	-0.313	
4		w	60		1			0.27	-1.313	
5	ʊ pai-	p	10			1		0.27	-1.313	
6		w	50		1			0.73	-0.313	
…	…	…	…	…	…	…	…	…		

As you can see, the log likelihood, −198.44, is a negative number, since the log of a probability (something less than 1) is always negative. Our goal is nevertheless to maximize this value — we are looking for a negative value as close as possible to zero.

Now that we have a cell that calculates log likelihood, we can use it as our accuracy metric, guiding the search for the best weights. This can be carried out in Excel using the "Solver" utility, which comes with the program but must be activated.[8] Unfortunately, although Solver programs exist for Google Sheets and Libre Office, at the time of this writing these versions do not contain the algorithm necessary for fitting a MaxEnt model.

To find the best weights, we recommend carefully following the following steps:

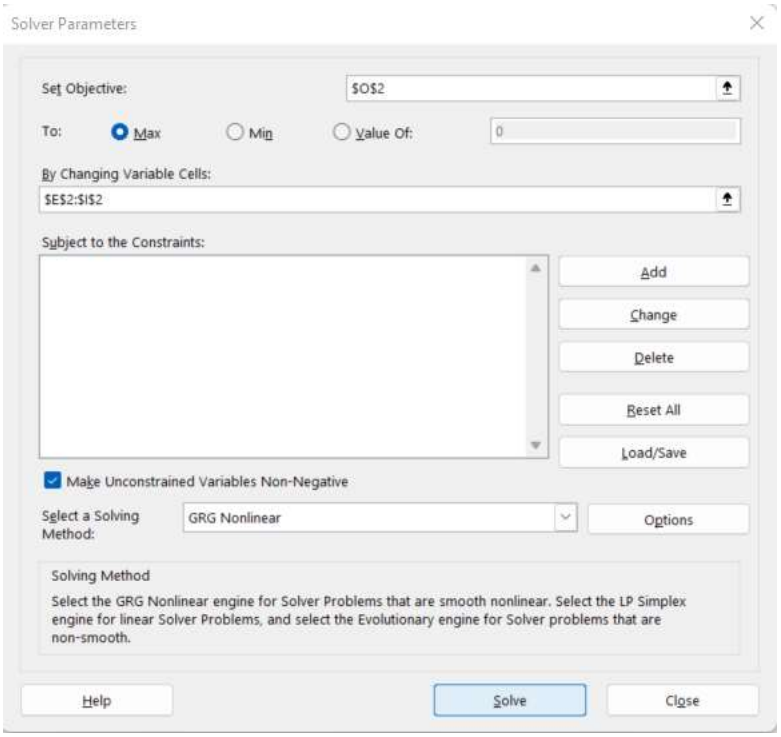
1) **Initialize:** Set all constraint weights to 0 (in K4.xlsx, this would be cells E2:I2).

[8] In the version of Excel we are currently using the procedure is: File > Options > Add-ins. In the Manage drop-down menu, select "Excel Add-ins" and click Go. Then, check the box next to "Solver Add-in" and click OK.

2) **Select Log likelihood as objective:** Open the Solver interface, and select the cell where you have calculated log likelihood (here, cell O2) for the "Set Objective" field. Since we are trying to maximize likelihood, you should select the "Max" button.
3) **Select weights:** Select the cells which contain your weights (cells E2:I2) in the "By changing variable cells" field.
4) **Keep weights non-negative:** In order to avoid negative weights, make sure that the checkbox next to "Make Unconstrained Variables Non-Negative" is checked.
5) **Select solving method:** Where it says "Select a solving method", choose "GRG Nonlinear" (usually the default).[9]

The settings of all these conditions for the Solver are shown in the screen clip of Figure 2.*22*

Figure 2: Launching the Solver on the K4 problem



With these conditions set, one can click on the Solve button, and for at least for schematic problems of the size described here, one finds that the algorithm it uses is able to find the best-fit weights very quickly. These weights are as follows:

[9] GRG Nonlinear in general terms can be studied in Lasdon et al. (1974), though the specific implementation in Solver is proprietary. GRG Nonlinear is just one of many algorithms that can be used to effectively compute MaxEnt weights; the choice of algorithm is in most cases a purely practical matter and will not affect the analytic results obtained.

(28) *Best-fit constraint weights for the K4 analysis*

IDENT(son)$_{content}$	13.37
AGREE(cont)	12.35
*p	3.46
IDENT(son)	5.03
AGREE-round	3.18

These weights provide a close fit to the observed probabilities in K4, as (29) shows.

(29) *Observed vs. predicted probabilities for the K4 analysis*

Input (see Fig. 1, column A)	*candidate*	*observed probability*	*predicted probability*
function word, after non-nasal consonant	p	0.829	0.828
	w	0.171	0.172
function word, after vowel	p	0.167	0.167
	w	0.833	0.833
content word	p	1.000	1.000
	w	0.000	0.000
function word, after nasal	p	1.000[10]	1.000
	w	0.000	0.000

Thus we now have a complete MaxEnt analysis of K4, which fits the data quite well. All steps in the creation of the analysis have been described explicitly, with the one exception of the GRG Nonlinear algorithm, internal to the Solver, that locates the sets of weights under which the objective function has been optimized; this alone has been left as a black box.

2.7 K5: Multiple perturbers

Having introduced the basics of MaxEnt grammar via the toy languages K3 and K4, we now turn to a final toy language, which is closely based on the real Khalkha data analyzed by Fuller (2023). This more complex example will illustrate a more intricate type of data pattern that shows up widely in MaxEnt analysis.

First, a trivial change: we will treat the post-nasal environment not as an absolute blocker of lenition, but merely a downward perturber; i.e., forms like /ʃitt͡ʃʰəxsə**n p**aisən/, from K4, do occasionally undergo lenition.

More fundamentally, in K5 we adopt a crucial observation made by Fuller, who found different rates of lenition for different function words. In *pi*, the first person pronoun, she observed almost no lenition (just 7 lenited forms out of 191). On the other hand for the particle *pol*, she observed 112 cases of lenition out of 163. She observed different rates in two other

[10] Note that these predicted probabilities are not exactly 1 and 0, but rather very close to 1 and zero; close enough that Excel displays only 1 or 0.

function words as well, one of which is homophonous with the particle *pol*. In this case we adopt for K5 some values actually taken from Fuller.

(30) *Data from Fuller, figure 10*

Function word:		*Instances with p*	*Lenited instances*	*Lenition rate*
pol	particle used for interrogatives, conditionals, and focus	51	112	68.7%
pol-	copula, "to become"	55	57	50.9%
pai	copula, "to be"	142	99	41.1%
pi	1st person pronoun	184	7	3.7%

Fuller found that these preferences of each function word coexist alongside the grammatical effects of blockers (the nasals in K4) and of the lenition-preferring environment (post-vowel in K4). Each function word, though it has its own baseline probability of leniting, lenites more in the lenition-preferring environment, and less in the blocking environment.

In the present case, morpheme identity is another type of perturber, in this case lexical rather than phonological. If, for convenience's sake, we think of the lenition rate for *pai* as a sort of default, then using the other morphemes *pi*, *pol*, or *pol-* in effect perturbs the lenition rate downward.

To give the full data set, we cross-classify the effects of phonological environment already seen with the effect of morpheme identity, as in (31). The counts are based on Fuller's actual corpus data, but simplified somewhat.

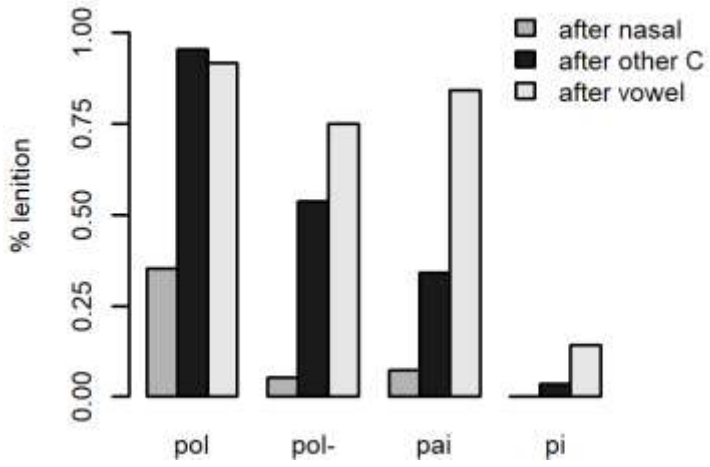
(31) *The K5 data pattern*

Function word	*UR type*	*Example*	*SR*	*Observed*
pol	... C [$_{\text{function}}$ p ...	[natə**t p**ɔɮ]	p	3
			w	65 (95.6%)
	... V [$_{\text{function}}$ p ...	[ɔrəɮt͡sʰ**ɔ p**ɔɮ]	p	2
			w	22 (91.7%)
	... [+nasal] [p ...	[ju**m p**ɔɮ]	p	46
			w	25 (35.2%)
pol-	... C [$_{\text{function}}$ p ...	[t͡ʃiɮ**t p**ɔɮsən]	p	31
			w	36 (53.7%)
	... V [$_{\text{function}}$ p ...	[xit͡sʰ**u p**ɔɮt͡ʃʰəxsən]	p	6
			w	18 (75.0%)
	... [+nasal] [p ...	[xu**n p**ɔɮəxig]	p	18
			w	1 (5.3%)

pai	... C [$_{\text{function}}$ p ...	[pɔttə**g p**aisən]	p	58
			w	30 (34.1%)
	... V [$_{\text{function}}$ p ...	[tʊt$^{\text{h}}$**ʊ p**aiɢa]	p	10
			w	53 (84.1%)
	... [+nasal] [p ...	[ʃ͡ɪtʃəxsə**n p**aisən]	p	75
			w	6 (7.4%)
pi	... C [$_{\text{function}}$ p ...	[pajərla**t p**i]	p	131
			w	5 (3.6%)
	... V [$_{\text{function}}$ p ...	[ətɔ **p**i]	p	12
			w	2 (14.3%)
	... [+nasal] [p ...	[tɛgsə**n p**i]	p	41
			w	0 (0%)

Note that while every function word has its own preference for lenition or not, every function word is still affected by the phonological environment. For all function words, the proportion of lenition after vowels is greater than elsewhere. Likewise, after nasals, the proportion of lenition is always lower. This can be seen clearly in Figure 3, which shows the proportion of lenition in each phonological environment for each function word. The light grey bars indicate the proportion of lenition after vowels, and are highest for every function word except *pol* where rates of lenition in the elsewhere context and after vowels are near ceiling. Likewise, the medium-grey bars, which indicate proportion lenition after nasals, are lowest for all function words.

Figure 3: Lenition rates in K5: intersecting effects of function word and phonological environment



What we see here in K5 is another, more complex, case of interacting pressures in the grammar. It is not enough to label some function words as exceptions. It would be tempting, for instance, to label *pi* as exceptionally non-leniting. But even *pi* lenites the most (percentage-wise) after vowels, and the least after nasals. Both individual lexical preferences and grammatical pressures jointly affect the likelihood of lenition.

To model this in MaxEnt, Fuller proposes lexically-indexed copies of the IDENT(son) constraint (Pater 2010), adapted to K5 as follows:

(32) *Lexically-indexed IDENT constraints for K5*

a. IDENT(son)$_{pol}$ Output correspondents of an input [αsonorant] segment in the lexical item *pol* are also [αsonorant].
b. IDENT(son)$_{pai}$ Output correspondents of an input [αsonorant] segment in the lexical item *pai* are also [αsonorant].
c. IDENT(son)$_{pol\text{-}}$ Output correspondents of an input [αsonorant] segment in the lexical item *pol*- are also [αsonorant].
d. IDENT(son)$_{pi}$ Output correspondents of an input [αsonorant] segment in the lexical item *pi* are also [αsonorant].

These lexically-indexed constraints act just like their general counterpart IDENT(son), but only assign violations within the lexeme to which they are indexed.

Fuller uses one lexically-specific version of IDENT(son) for each function word in her data, each with a different weight. Because they are all versions of IDENT(son), they will all conflict with *p. Their weight relative to *p reflects that function word's overall propensity for lenition.

The tableau in (33) shows the lexically-specific constraint analysis for K5. The four IDENT(son) constraints assign violations identically, but apply to different inputs, only assigning violations when their indexed lexical item is present in the input. (For brevity we leave out general IDENT(son).) The lexically-indexed IDENT(son) constraints interact with the general phonological constraints from our K4 analysis: *p, AGREE(cont), and POSTVOCALICLENITE.

(33) *Tableau for K-5*

word	context	cand	obs.	AGREE(cont) 2.71	*p 2.23	POSTVOCLENITE 1.63	Id(son)$_{pol}$ 0.00	Id(son)$_{pai}$ 2.59	Id(son)$_{pol-}$ 2.22	Id(son)$_{pi}$ 5.55	H	p	Obs. p
pol	elsewhere	p	3		1						2.23	.10	.04
		w	65				1				0.0	.90	.96
	after V	p	2		1	1					3.86	.02	.08
		w	22				1				0.0	.98	.92
	after nasal	p	46		1						2.23	.62	.65
		w	25	1			1				2.71	.38	.35
pol-	elsewhere	p	31		1						2.23	.50	.46
		w	36						1		2.22	.50	.54
	after V	p	6		1	1					3.86	.16	.25
		w	18						1		2.22	.84	.75
	after nasal	p	18		1						2.23	.94	.95
		w	1	1					1		4.93	.06	.05
pai	elsewhere	p	58		1						2.23	.59	.66
		w	30					1			2.59	.41	.34
	after V	p	10		1	1					3.86	.22	.16
		w	53					1			2.59	.78	.84
	after nasal	p	75		1						2.23	.96	.93
		w	6	1				1			5.30	.04	.07
pi	elsewhere	p	131		1						2.23	.97	.96
		w	5							1	5.55	.03	.04
	after V	p	12		1	1					3.86	.84	.86
		w	2							1	5.55	.16	.14
	after nasal	p	41		1						2.23	.998	1
		w	0	1						1	8.26	.002	0

In the second row of tableau (33) we see the fitted weights obtained using the log likelihood criterion with the Solver, as detailed in §2.6.4 above. The final column shows the probabilities predicted by the analysis. The detailed calculations can be viewed in the **K5.xlsx** spreadsheet in the Supplementary Materials.

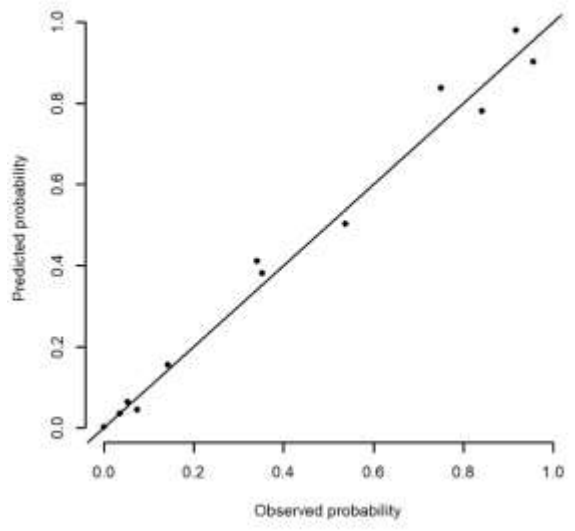
The reader may have noticed that the weight of Id(son)$_{pol}$ came out as zero. Oddly, this turns out to be no cause for alarm: since every [w] violates precisely one member of the IDENT(son) family, it turns out that all the matters is the *relative* weights within this family. For discussion, with the relevant math, see §4.5.3 below.

2.7.1 Assessing the K5 analysis with scatterplots

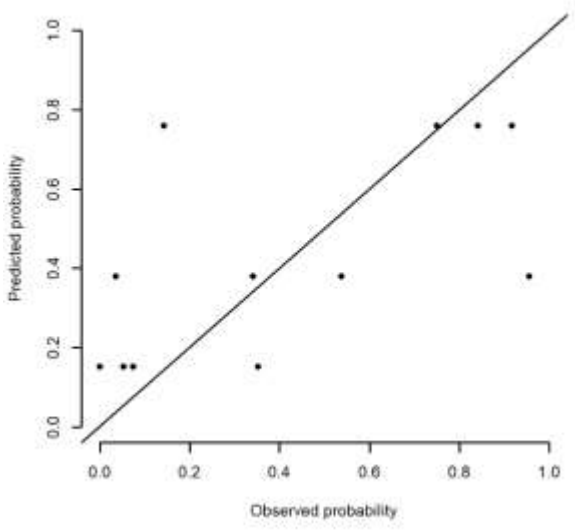
With K5, we have enough data that it makes sense to visually compare the observed and predicted values in a scatterplot. We compare the model probabilities of each candidate with the fraction of observations that that candidate receives (that is, observed cases divided by total cases for that input.) The left side of Figure 4 shows observed values for the lenition candidates on the *x* axis, and the corresponding predicted values on the *y* axis. For purposes of comparison, the graph also contains the line $y = x$. If the model's fit were perfect, every point would be on this line. Indeed, we can see that all the points are very close to the line, indicating a good fit of our model to the data.

Figure 4: Comparing "observed" with predicted values for MaxEnt model of K5

a. *Full model (from (33))*



b. *Comparable model lacking lexically indexed constraints*



For purposes of comparison, we also give on the right the scattergram for a model that is similar, but which replaces the four lexically-indexed constraints with one single IDENT(son) constraint (see K5.xlsx/K5_noIndexation, a sheet within K5.xlxs, for weights and computations). For this model, adherence of the points to the $y = x$ line is patently less close; it really did make sense to split up IDENT(son) into morpheme-specific versions. For more on model comparison and the use of scattergrams, see Chapter 5 below.

2.7.2 K5 vs. real Khalkha

Although our K5 dataset is close to real Khalkha, we wish to briefly point out a few important differences, referring to Fuller (2023). The biggest difference is that in Khalkha, /p/ can actually surface three different ways: as [p], as [w], and as null. This necessitates richer system of Faithfulness constraints. A second difference is that we have simplified the nature of

the phonological perturbers. In real Khalkha, curiously enough, they seem to form non-natural classes (cf. Mielke 2005). Lenition is dispreferred after nasals [m], [n], and [ŋ] (as in K-5) but also after [ɮ] and [w]. The lenition-preferring environment is not postvocalic, but the set consisting of velars and uvulars: [g], [ɢ], [x], plus the diphthong [ui]. With Fuller, we hope that further research will shed light on why these sounds pattern together.

2.8 Visualizing the K-5 pattern with displaced sigmoids

At this point we seek to understand the K5 analysis more deeply, particularly how the constraint weights result in the observed patterns of candidate probability.

2.8.1 Examining the effects of the weight of one single constraint

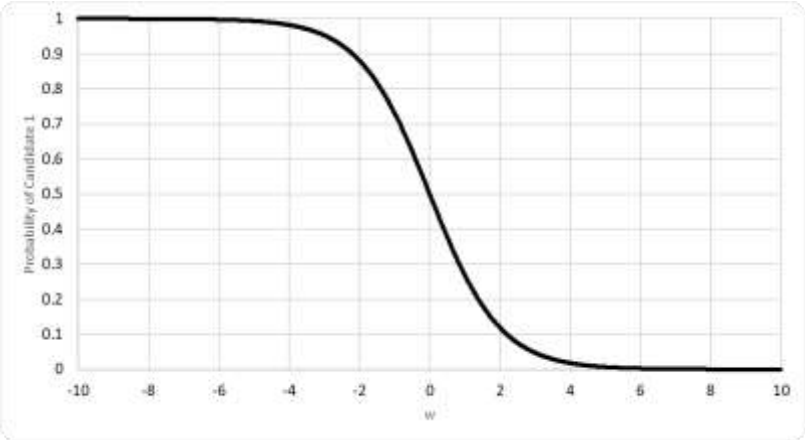
We first examine, in its most primal form, the relationship between a constraint weight and candidate probability. For this purpose imagine a one-constraint grammar in which there are two candidates, one that violates the single constraint C, and one that has no violations at all; hence tableau (34):

(34) *A minimal tableau for illustrating the effect of a constraint weight*

	C				
weight:	*w*				
/Input/		Harmony	$e^{-Harmony}$	Z	p
Candidate 1	1	x	e^{-w}	$1+ e^{-w}$	e^{-w} / Z
Candidate 2		0	1	$1+ e^{-w}$	$1 / Z$

The tableau in (34) is done with formulae, not values, so we can use it to check how the probability of Candidate 1 varies over different values of the weight *w*. Here, it will be conceptually helpful to examine even those values of *w* that are negative, contrary to common analytical practice. Plotting *P*(Candidate 1) against the weight *w*, we get the curve shown in Figure 5:

Figure 5: Single-constraint grammar: How P(Candidate 1) *varies with w*



This curve is called a **sigmoid**, meaning S-shaped. Looking at Figure 5 in detail, we can observe the following. (a) At $w = 0$, the probability of Cand 1 is 0.5. This makes sense, because this is the equivalent of a constraintless grammar, and such a grammar should assign 50-50 probability to the two candidates. (b) The curve is *monotonic*, in that higher values of *w* always result in lower values for *P*(Candidate 1). All differences in *w* will result in differences in *p*. (c) However, as we increase *w* going outward from zero, we observe *diminishing returns*: each additional unit of weight has successively less influence. (d) Probability never actually reaches zero, though the value of *P*(Candidate 1) for high values of *w* becomes infinitesimally small. (e) In the non-standard region of negative weights, on the left side, the constraint C acts as a reward, rather than a penalty, and the curve rises going leftward, again with diminishing returns and asymptoting toward 1. (f) The two sides are mirror-wise identical: a MaxEnt sigmoid is always symmetrical around the location of 50% probability.

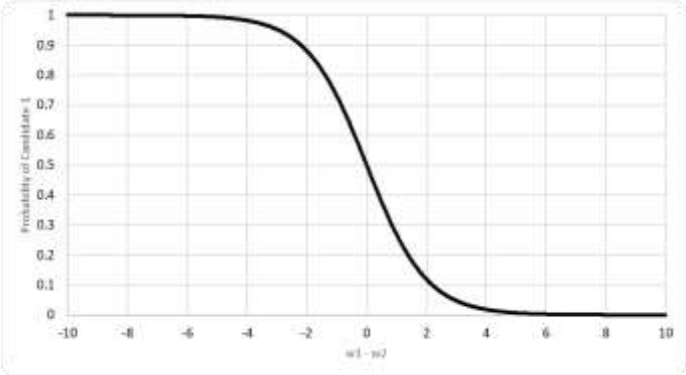
Let us next consider a more realistic example, involving constraint conflict. We suppose that instead of one constraint C we have two constraints C1 and C2, which are opposed to each other. A sample tableau for such a case is in (35).

(35) *A minimal tableau for illustrating the effect of conflicting constraints*

	C1	C2				
weight:	*w1*	*w2*				
/Input/			Harmony	$e^{-Harmony}$	Z	p
Candidate 1	1		w1	e^{-w1}	$e^{-w1} + e^{-w2}$	e^{-w1} / Z
Candidate 2		1	w2	e^{-w2}	$e^{-w1} + e^{-w2}$	e^{-w2} / Z

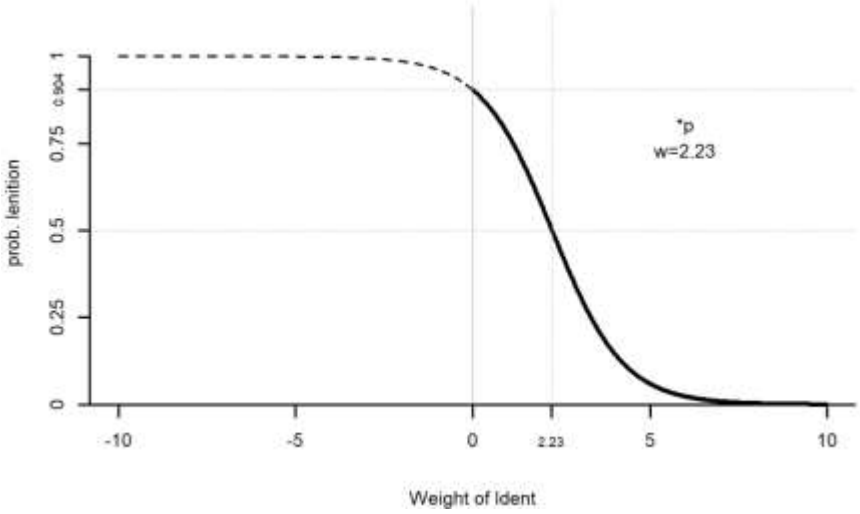
Here, the appropriate analytical focus is on the *difference* between the weights, $w_1 - w_2$. The sigmoid that we get is identical to the one in Figure 5, except that the meaning of the *x*-axis has changed, $w_1 - w_2$ rather than *w*.

Figure 6: Two-constraint grammar: How P(Candidate 1) *varies with the weight difference*



Notice that a graph rather like Figure 6 could be plotted for a subset of our Khalkha data; one example is for the first input in the tableau (33) for K5, where the output probabilities are determined exclusively by the two conflicting constraints *p and IDENT(son)$_{pi}$. If we fix the weight of *p at 2.23 and explore what we get when we vary the weight of IDENT(son)$_{pi}$, then the curve comes out as in Figure 7.

*Figure 7: Sigmoid curve: result of varying weight of IDENT(son)$_{pi}$ when weight of *p is constant*



As can be seen, if the weight of IDENT(son)$_{pi}$ is set very high, lenition is predicted to be very unlikely. If IDENT(son)$_{pi}$ is given a weight identical to that of *p, i.e. 2.23, the rate of lenition is predicted to be 50%; this is indeed the point where the curve crosses the 50% line. As the weight of IDENT(son) approaches zero, the probability of lenition goes up; with $w_{\text{IDENT(son)}} = 0$, it is 90.4%. Lastly, were we to permit negative weights, we could generate a lenition probability as close to 1 as we wish; this region is shown in the dotted portion of the sigmoid.

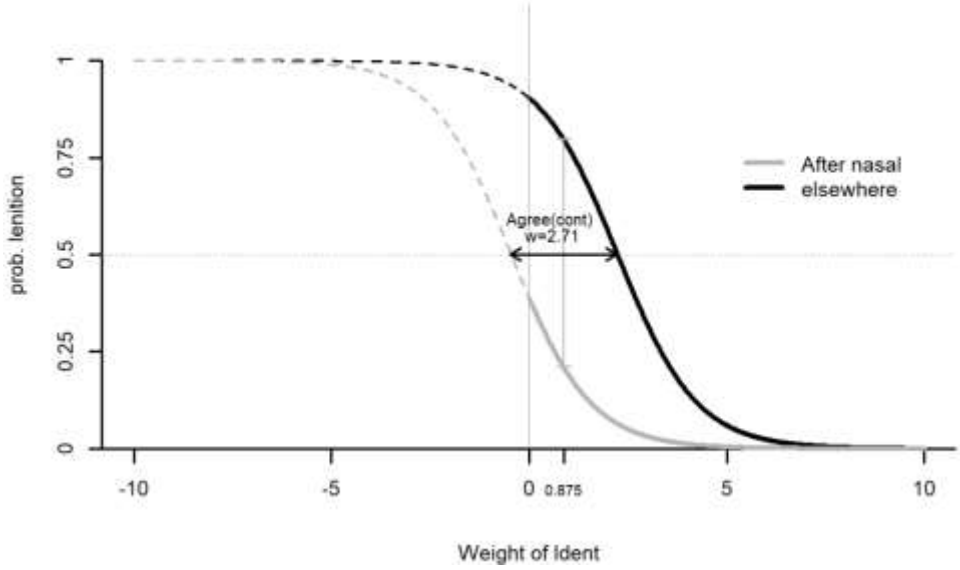
In more complicated cases we would actually want to plot the difference in *Harmony* between the two candidates, rather than the different in weights; see for instance §3.2.4 below. In Figure 7, the two approaches come out the same because there is exactly one violation per candidate.

2.8.2 The sigmoids arising from perturber constraints

Let us now consider what sort of sigmoid patterns we get when we introduce perturber constraints into the analysis. Here, we will plot two sigmoids: one for inputs that violate the perturber constraint, and one for inputs that do not. What emerges when we do this — and follows from the MaxEnt math — is that the two sigmoids are *identical and spaced apart by the weight of the perturber constraint*. We will call this configuration **displaced sigmoids**.

As an example of displaced sigmoids, consider K5, where AGREE(continuant) acts as a perturber constraint, lowering the probability of lenition after nasals. Accordingly, in Figure 8, we see a second sigmoid in gray, which plots the probability of lenition for inputs where /p/ follows a nasal (so that the lenited candidate violates AGREE(continuant)). This second sigmoid is derived from activity of all three constraints, *p and AGREE(continuant) (whose weights are held constant); and IDENT(son)$_{pi}$, whose weight changes according to the *x* axis. As shown by the arrow, the two sigmoids are spaced apart on the *x* axis by 2.71 units, which is, in fact, the weight of AGREE(continuant).

Figure 8: An example of displaced sigmoids: identical sigmoids spaced apart by the weight of AGREE(continuant)

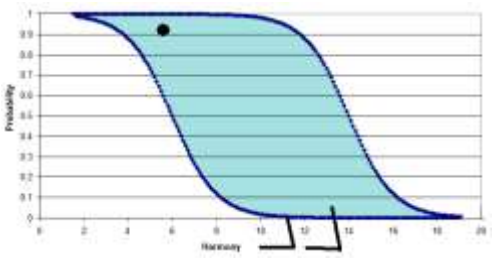


Of course, on the *vertical* axis, the distance between the two sigmoids varies considerably. In the present case, the maximum distance is 0.587, occurring where the weight of IDENT(son)$_{pi}$ is 0.875 (see thin vertical line in Figure 8). This distance approaches zero for ever higher or lower weights of IDENT(son)$_{pi}$. What this means is that the effect of a perturber constraint is *contextually determined*: it is always maximal at the point halfway between where the two sigmoids cross 50%, and is ever smaller going off in either direction.

The contextual variation in the effect of a perturber is a fundamental prediction of MaxEnt, and is repeatedly referred to in analytical work.

A useful mnemonic for these contextual effects comes from a jocular nickname attached to the displaced-sigmoid pattern:[11] the **wug-shaped curve**. We can see the shape of a wug (the cute mythical creature first drawn by Berko 1958) by replotting Figure 8 with a somewhat higher value for w_{Ident} and adding appropriate anatomy.

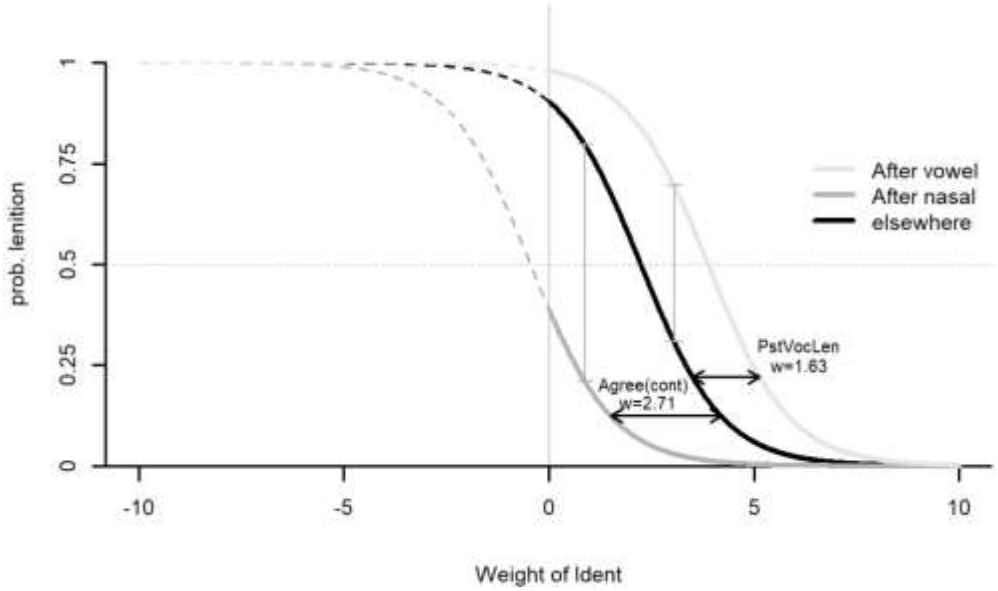
Figure 9: The wug-shaped curve



The mnemonic is: that the effect of a perturber constraint is strong in the belly of the wug, weak in beak and tail.

We consider next the case in which there is more than one perturber constraint. Here, there will be three sigmoids, one for the baseline condition and a displaced sigmoid for each perturber. For instance, in Figure 10 we add a second perturber from K5, POSTVOCALICLENITE. This constraint favors lenition postvocalically, so its sigmoid is displaced rightward.

Figure 10: Sigmoid curves for K5 with two perturbers

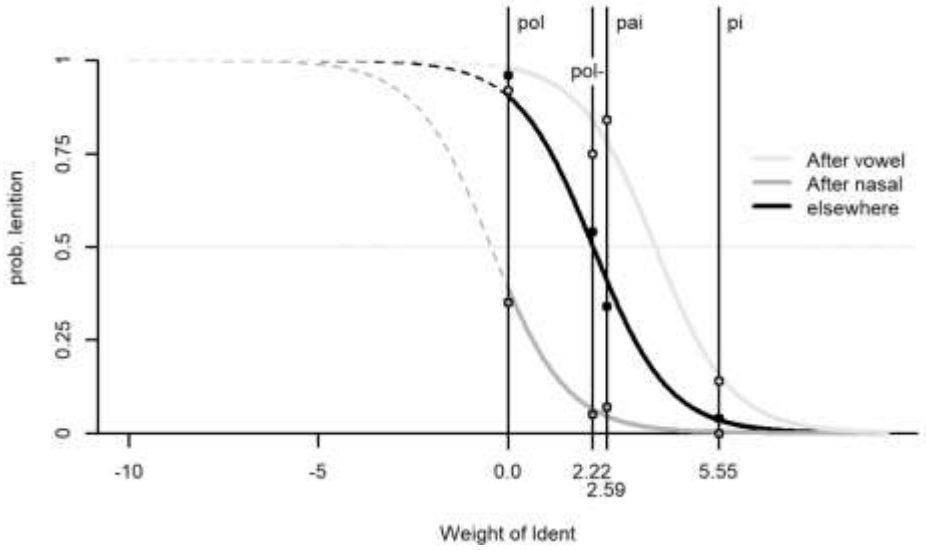


So far, we have dealt with a hypothetical case in which the weight of IDENT(son)$_{pi}$ is varied freely for illustrative purposes. We next attempt to better understand the actual K5 analysis using the same technique. The sigmoid curves will be identical to those of Figure 10, but we will mark with a vertical line the actual fitted weight for IDENT(son)$_{pi}$, which is 5.55. We will also include

[11] The name was suggested by Dustin Bowers.

vertical lines marking the weights of the other three morpheme-indexed IDENT(son) constraints, namely IDENT(son)$_{pi}$, IDENT(son)$_{pol}$, and IDENT(son)$_{pol\text{-}}$. The points at which these vertical lines intersect the sigmoid curves represent the predictions of our MaxEnt grammar. The figure also includes dots that show the (imagined) empirical values for the corresponding points in the K5 data. If our fitted model is a good one, then these dots should fall fairly close to the intersection points of the corresponding vertical lines and sigmoids.

Figure 11: Assessing the K5 model using displaced sigmoids



We see that the data points match fairly well with their predicted values. The largest errors are evidently those for *pol-* and *pai* in the postvocalic context, which are 0.09 and −0.06, respectively.

The use of annotated displaced-sigmoid diagrams like Figure 11 provides a useful complement to the use of scattergrams (§2.7.1) in evaluating an analysis, for it indicates how the predictions emerge from the constraints and their weights.

Beyond its role in assessing the accuracy of the analysis, the graph in Figure 11 makes a far more general point, namely the very fact that the data can be fitted with a set of displaced sigmoids — visually, the data are "wug-shaped". For instance, for the function word *pi*, the high weight of IDENT(son)$_{pi}$ means that lenition is, in general, fairly unlikely for this word. Accordingly, the effects of the perturbers are small. This is the tail of the wug. In contrast, for *pai*, *pol*, and *pol-*, the IDENT(son) constraints bear only modest weights. Thus, we are in the belly, and the perturbers have substantial effects.

2.9 Very high weights and the approximation of zero

We return for a moment to K4, a language in which some outcomes are obligatory (21). For instance, lenition in K4 is completely blocked in the post-nasal environment. In our MaxEnt analysis, this outcome is attributed to a powerful constraint, AGREE(cont), whose best-fit weight is 12.35. For similar reasons, the weight of IDENT(son)$_{\text{content}}$ is also very high, 13.37.

These weights must be interpreted with care. First, we note that in fact there is *no* weight assignment that will fit the empirical value of zero exactly. To see this, consider the stage of the MaxEnt calculations given in (16b), where eH is calculated using eH = e^{-H}. No matter how high is the value of H (resulting from high constraint weights), e^{-H} will always be greater than zero. So the situation is akin to taking a limit: the higher the weight we assign to AGREE(cont) and IDENT(son)$_{\text{content}}$, the closer to zero will be the probability assigned to zero-frequency candidates like *[ʃit-t͡ʃʰəx-sən **w**ai-sən] and *[ɔrʊɮ-əx **w**ɔɮəm-t͡ʃʰɔi]. But the value will never actually reach zero; we can only approximate it. The ideal weight for AGREE(cont) and IDENT(son)$_{\text{content}}$ is, in this sense, infinity.

In the real world, weights are computed by an algorithm that labors to find ever better approximations to the observed distribution. The algorithm will always include one or more parameters that tell it when to stop trying.[12] The values obtained for constraints like AGREE(cont) and IDENT(son)$_{\text{content}}$ are the values that have been computed up to this stopping point.

There are two lessons here. One is a simple practical one: for some constraints, it is sensible for the analyst *not to obsess on the exact weights* when they are high, even where they differ substantially. All such values are more or less arbitrary stopping points for the search algorithm, the points where its settings lead it to accept the current (tiny) predicted value for zero-frequency candidates as close enough. Perhaps surprisingly, in MaxEnt, the difference in weights between (say) 20 and 50 can be very unimportant, whereas the difference between 1 and 2 might be quite important.

A second lesson goes a bit deeper. Anyone one who uses MaxEnt must be prepared to accept an infinitesimally small number as the practical equivalent of zero. For instance, our K4 grammar assigns the probability 3.2×10^{-7} to the bad candidate *[ɔrʊɮ-əx **w**ɔɮəm-t͡ʃʰɔi] and 8.9×10^{-7} to the bad candidate *[ʃit-t͡ʃʰəx-sən **w**ai-sən]; these were shown earlier in (18) for convenience as "≈ 0". The difference between these numbers and zero is smaller than the measurement error for almost all data sets.

Lastly, we point out the class of constraints for which the "infinite weight" syndrome arises: typically they are constraints that *prefer a winner*, but never *prefer a loser*, in the terminology of (Prince 2002b). In classical Optimality Theory, such constraints are the ones that can always safely be ranked at the top of the grammar. In MaxEnt, they have idealized infinite weights.

2.10 Some Excel tips

Since this is a handbook, we offer a brief section on how one can use Excel more quickly and reliably to carry out MaxEnt calculations.

2.10.1 Avoiding weight crashes

The fact that in some cases a constraint's ideal weight is infinite can raise practical problems in Excel. Solver's weight-fitting algorithm will halt after a certain amount of computation, often

[12] To alter the stopping parameter in the Excel Solver, click Options, GRG Nonlinear, and change the value in the box labeled Convergence.

landing on a weight that is high enough to give a good fit to the data. However, in some cases Solver will fit a constraint weight which predicts probabilities so low that Excel cannot represent them. In this case, the algorithm will crash, and cells which cannot be represented will yield the #NUM! error message instead of actual values.

We discuss a principled way for dealing with this problem in §4.7.4 below, but here give an ad hoc solution that does not require much additional calculation. Specifically, it is possible in the Solver simply to impose a hard upper limit on weights. A maximum of 50 is often sufficient to produce an accurate analysis while avoiding crashes. To implement such a limit, do the following:

(36) *An ad hoc procedure for limiting weights*

- In the Solver window, click "Add" to the right of the box labelled "Subject to the Constraints".
- In the left hand field, select the cells which contain the weights.
- Select "<=" for the middle drop-down menu, and type "50" in the right hand field.

The Solver will then give the best-fit answer available, subject to this weight limit: weights that otherwise would be unmanageably large will be limited to 50. For specific problems you may find that an upper limit greater than 50 is necessary, or a limit lower than 50 is sufficient.

2.10.2 A safer method for calculating Z

A common source of error in a MaxEnt spreadsheet, often hard to detect, arises in the calculation of the Z values. These must be obtained by summing the eH values (cf. (16b)) for *exactly* the set of candidates for a particular input. Where each input has the same number of candidates, it is not hard to avoid error, but if the inputs can have different numbers of candidates it is easy for a slip of the mouse to create an error. Here is a safer, if more verbose method.

First, you must include (perhaps just adding it in) a column in which the input is specified for every row. For instance, for the first two inputs of (24) we need a column like column B in (37).[13]

[13] The column B is easy to create from the old column B seen in (24) by adding a new column (call it column C) and using a formula like `=if(C3="",B2,C3)` which extends what is already in B unless a new input appears in the copies the new column when it contains a new input, else extends what is already in B.

(37) *A method for greater reliability in calculating Z*

Excel column names

row #	**B**	C	D	...	H	I	J	K	**L**	M
1	**Example**			...	Id(son)	Postvoc Lenite	Harmony	eH	**Z**	p
2		output	count		4.00	2.00				
3	**g poɣən**	p	290				=SUMPRODUCT(E$2:I$2, E3:I3)	=EXP(−J3)	**=SUMIF(B:B, B1, K:K)**	=K3/L3
4	**g poɣən**	w	60		1		=SUMPRODUCT(E$2:I$2, E4:I4)	=EXP(−J4)	**=SUMIF(B:B, B2, K:K)**	=K4/L3
5	**ʊ pai-**	p	10			1	=SUMPRODUCT(E$2:I$2, E5:I5)	=EXP(−J5)	**=SUMIF(B:B, B3, K:K)**	=K5/L5
6	**ʊ pai-**	w	50		1		=SUMPRODUCT(E$2:I$2, E6:I6)	=EXP(−J6)	**=SUMIF(B:B, B4, K:K)**	=K6/L5
…	…	…	…	…	…	…	…	…	…	…

Once you have a column in place that repeats the inputs, then the reliable calculation of Z can be based on the formula `=SUMIF(A:A, A4, J:J)`,[14] which can be entered once into the first cell of the Z column (in (37), cell A4), then copied down the entire column. This formula takes the subset of J (the eH column) where A is equal to the value in A4 (or whatever row you are on), even if it is not contiguous, and sums them. We recommend using this method when you have different numbers of candidates for different inputs, making it dangerous to compute all the Z calculations with separate copy-paste moves. It is also helpful if you plan on experimenting with changing the numbers of candidates for each input.

2.10.3 Troubleshooting

Sometimes a first effort to solve a phonological problem yields poor results, such as a bad fit to the data or nonsensical constraint weights (such as zero for a patently necessary constraint). We offer some hints for troubleshooting.

The problem may be with the linguistic analysis. For techniques for fixing shortcomings in this area see §5.2 below. However, the problem may lie with the Excel implementation. The following checks may be helpful for such troubleshooting, or indeed, you might find it useful to do them before you ever click the Solve button.

(a) Check that all the constraint violations in your spreadsheet have been correctly assigned.

[14] A caution: this formula is not case-sensitive. If you have inputs that differ solely in capitalization, use instead `=SUMIF(--EXACT(A4,A:A)*J:J)`.

(b) Visual spot-check: Check a few Harmony values at the beginning, middle and end of your spreadsheet by hand. The most common error here is that the Harmony column has left out a constraint; D3:G3, for example, when it should be D3:H3. If you set all the weights by hand to the value 1, it becomes easier to do this check.

(c) If you did *not* use the method given in the previous section, then hand-check your Z values to make sure that they do indeed sum over all the candidates for the same input.

(d) Check each input's probability distribution, to make sure it sums to 1. For small spreadsheets, you can highlight the candidates for the input with your mouse or tracker and check the sum at the bottom of the Excel screen. To be more thorough, you can use the following formula in cell M4 (in the case of (37)), and copy it down the column: `=IF(SUMIF(A:A,A3,L:L)=1, "", "!")`. Then, search for "!" in that column, which will indicate an output probability distribution that did not sum to 1. If you find an error it could mean that, perhaps, your Z scores were not calculated correctly, or perhaps that your exponentiation is off — perhaps you didn't include the negative sign.

(e) Finally, go through each of your constraints and manually change the weight. You should see the probabilities in column L change for all inputs in which some candidate violates that constraint.

(f) It is especially helpful to carry out the above spot checks when you add a new input or candidate to your spreadsheet, which we have found to be a frequent source of error.

(g) To obtain consistent results when revising your analysis, it is recommended that you reset the weights to zero before launching the Solver. The benefit of this is that you are more likely to obtain a minimum-weight solution, in which useless constraints will stand out because they are assigned a weight of zero. For a more principled way of finding the minimum-weight solution, see §4.7.4.

(h) In larger simulations, the termination criterion employed as default by the Solver may not permit the most accurate feasible solution to be found. In our experience, clicking the Solver button a second time often offers modest improvement in log likelihood. It seems unnecessary ever to click a third time.

2.11 More MaxEnt software

We mention two other forms of software that can implement a MaxEnt grammar and find the best-fit weights.

The **MaxEnt Grammar Tool** is available at brucehayes.org/MaxentGrammarTool/. A number of papers in the literature have used this software to implement a MaxEnt grammar. While it works perfectly well, we currently favor the Excel/Solver combination over it, since Excel renders more transparent the work that has been done.

Maxent.ot is a software package usable in the well-known statistical analysis system R. For documentation, see Mayer et al. (2024). It requires the user to have some fluency in writing R

scripts, and perhaps is thus less suitable for beginning work. For a use that we made of this package, see §7.5.3. A key advantage of Maxent.ot is that it lets you embed MaxEnt optimization into larger routines. We suspect Excel can handle slightly larger problems and is a bit faster (i.e., on a single run).

2.12 For Further Reading

We cover here only some works relevant to this specific chapter, passing over important papers that bear more directly on individual chapters to follow.

2.12.1 MaxEnt as a descendant of OT

To become an informed user of MaxEnt, we feel it is helpful to know a lot about the literature of Optimality Theory. Three excellent textbooks that have been written for OT are Kager (1999) and McCarthy (2002, 2008). While MaxEnt rejects the ranking principle of OT as a basis for candidate selection, nevertheless a great many research questions about MaxEnt originate with, and continue to be, research questions about OT.

2.12.2 MaxEnt as a descendant of Classical Harmonic Grammar

Pater (2009) is a paper about classical Harmonic Grammar (§2.5.1), the nonstochastic version of MaxEnt. It makes useful arguments about the relative generative capacity of Harmonic Grammar as compared to Optimality Theory.

2.12.3 The research tradition in variationist linguistics

Perturbers were, as far as we know, first noticed in the variationist research tradition centered on William Labov. Two seminal papers are Labov (1969) and Cedergren and Sankoff (1974), and a helpful text is Walker (2010). MaxEnt math entered the field with Rousseau and Sankoff (1978), where it served to augment the phonological theory of Chomsky and Halle (1968). In this "variable rules" approach, each rule comes with a little MaxEnt grammar with just two candidates, "Apply rule" and "Don't apply rule". The framework proved influential and inspired a large literature, notably work that applied Sankoff's "Varbrul" software.

2.12.4 Origin of the MaxEnt framework

MaxEnt as a framework arose in the early years of this century when computer scientists noticed that mathematical tools that they were already using could be bolted onto OT to form an effective stochastic theory. The key reference, from which the name MaxEnt is taken, is Goldwater and Johnson (2003). Another early paper with extensive tutorial material is Hayes and Wilson (2008).

2.12.5 Analysis with MaxEnt

Most papers written in modern linguistics have a theoretical point to make, but in the present context it can be worth examining them simply as case studies. Here are a few papers that we feel would be particular useful to novice practitioners of MaxEnt.

(38) *Some analytic papers using MaxEnt*

Paper	*Topic*
Hayes and Londe (2006)	Hungarian vowel harmony
Zhang et al. (2011)	Tone Sandhi in Taiwanese
Linzen, Kasyanenko and Gouskova (2013)	Vowel-zero alternations in Russian prepositions
Jun (2015)	Korean /n/ insertion
McPherson and Hayes (2016)	Tommo So vowel harmony and morphological structure
Garcia (2019)	weight and stress in Portuguese
McCollum (2019)	Kazakh vowel harmony
Moore-Cantwell (2020)	English stress assignment; using MaxEnt to justify a novel constraint
Smith and Pater (2020)	Schwa-zero alternations in French

2.12.6 Displaced sigmoids

See Zuraw and Hayes (2017), Hayes (2017), Kawahara (2020, 2021).

References

Albright, Adam, and Bruce Hayes (2003) Rules vs. analogy paper xxx *Cognition*

Alderete, John (2003) Structural disparities in Navajo word domains: A case for LEXCAT-FAITHFULNESS. *The Linguistic Review*, *20*(2-4), 111-157. https://doi.org/10.1515/tlir.2003.006

Anttila, Arto (1995) xxx his dissertation

Anttila, Arto (1997) Deriving variation from grammar: A study of Finnish genitives. In *Variation, change and phonological theory*, ed. Frans Hinskens, Roeland van Hout, and Leo Wetzels. Amsterdam: John Benjamins.

Bailey, Charles-James N. (1973) *Variation and linguistic theory*. Washington: Center for Applied Linguistics.

Bakovic, Eric. (2000) *Harmony, Dominance, and control.* PhD Dissertation, Rutgers University.

Bayley, Rxxx. (2002) The quantitative paradigm. In Jack K. Chambers, Peter Trudgill, & Natalie Schilling-Estes (eds.), *The handbook of language variation and change* (pp. 117–141). Blackwell.

Becker, Michael (2009) xxx

Becker, Michael et. al. (2011) xxx

Becker, Michael and Maria Gouskova (2016). Source-oriented generalizations as grammar inference in Russian vowel deletion. *Linguistic inquiry*, *47*(3), 391-425.

Beckman, Jill (1997). Positional faithfulness, positional neutralization, and Shona vowel harmony. *Phonology* **14**. 1–46.

Bod, Rens, Jennifer Hay, and Stefanie Jannedy, eds. (2003) *Probabilistic Linguistics*. Cambridge: MIT Press.

Boersma, Paul (1998) *Functional Phonology. Formalizing the interactions between articulatory and perceptual drives*. Ph.D. dissertation, University of Amsterdam. The Hague: Holland Academic Graphics.

Boersma, Paul and Bruce Hayes (2001). Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32: 45-86.

Boersma, Paul and Joe Pater (2016). Convergence properties of a gradual learning algorithm for Harmonic Grammar. In John McCarthy and Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*. London: Equinox Press

Casali, Roderic F. (1997) Vowel elision in hiatus contexts: Which vowel goes? *Language*, *73*(3), 493–533. https://doi.org/10.2307/415882

Cedergren, Henrietta J., and David Sankoff (1974) Variable rules: Performance as a statistical reflection of competence. *Language* 50:333-355.

Chambers, Jack K., Peter Trudgill, and Natalie Schilling-Estes, eds. (2013) *The handbook of language variation and change*. Oxford: Wiley-Blackwell.

Chomsky, Noam and Morris Halle (1968) *The Sound Pattern of English*. New York: Harper and Row.

Coetzee, Andries W., and Shigeto Kawahara (2013) Frequency biases in phonological variation. *Natural Language and Linguistic Theory* 31:47-89.

Coetzee, Andries W., and Joe Pater (2008) Weighted constraints and gradient restrictions on place co-occurrence in Muna and Arabic. *Natural Language & Linguistic Theory* 26:289-337.

Cole, Jennifer and Lauren Trigo. 1989. Parasitic harmony. In Harry van der Hulst & Norval Smith (Eds.), Features, Segmental Structure and Harmony Processes. Foris, Dordrecht, pp. 19-38.

Culbertson, Jennifer, Paul Smolensky, and Colin Wilson (2013) Cognitive biases, linguistic universals, and constraint-based grammar learning. *Topics in Cognitive Science* 5:392–424.

Eisner, Jason (2000) Review of Kager: "Optimality Theory". *Computational Linguistics* 26:286–290.

Ernestus, Mirjam and R. Harald Baayen (2003) Predicting the unpredictable: interpreting neutralized segments in Dutch. *Language* 79:5–38.

Fuller, Corrina (2023) Word-initial /p/ in Khalkha Mongolian: Variation in connected speech. M.A. thesis, Department of Linguistics, UCLA. https://www.proquest.com/docview/2986554543

Garcia, Guilherme D. (2019). When lexical statistics and the grammar conflict: Learning and repairing weight effects on stress. *Language* 95(4):612–641.

Gelman, A., Su, Y., Yajima, M., Hill, J., Pittau, M. G., Kerman, J., & Zheng, T. (2010). Package 'arm'. Available at:http://cran.r-project.org/web/packages/ar.

Goldwater, Sharon and Mark Johnson (2003) Learning OT constraint rankings using a Maximum Entropy model. *Proceedings of the workshop on variation within Optimality Theory, Stockholm University*, 2003.

Gurevich, Naomi. (2011). Lenition. In M. van Oostendorp, C. Ewen, E. Hume, & K. Rice (Eds.), The Blackwell companion to phonology. New York: Blackwell Publishing.

Guy, Gregory R. 1993. The quantitative analysis of linguistic variation. In Denis Preston (ed.), *American Dialect Research*. Amsterdam: John Benjamins, 223–49.

Harris, James (1969) *Spanish phonology*. Cambridge, MA: MIT Press.

Hayes, Bruce (2008) *Introductory phonology*. Malden, MA: Wiley-Blackwell.

Hayes, Bruce (2016) Comparative phonotactics. *Proceedings of the 50th meeting of the Chicago Linguistic Society*, 265-285.

Hayes, Bruce (2017) Varieties of Noisy Harmonic Grammar. In Karen Jesney, Charlie O'Hara, Caitlin Smith and Rachel Walker (eds.), *Proceedings of AMP 2016*.

Hayes, Bruce (2022b) Deriving the Wug-shaped curve: A criterion for assessing formal theories of linguistic variation. *Annual Review of Linguistics* 8:474-494.

Hayes, Bruce and Zsuzsa Londe (2006) Stochastic phonological knowledge: the case of Hungarian vowel harmony. *Phonology* 23:59-10.

Hayes, Bruce and Colin Wilson (2008) A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry* 39:379–440.

Hayes, Bruce, Kie Zuraw, Peter Siptar, and Zsuzsa Londe (2009) Natural and unnatural constraints in Hungarian vowel harmony. *Language* 85: 822-863.

Jäger, Gerhard (2007) Maximum entropy models and stochastic Optimality Theory. *Architectures, rules, and preferences. Variations on themes by Joan W. Bresnan*, ed. by Annie Zaenen, Jane Simpson, Tracy Holloway King, Jane Grimshaw, Joan Maling, and Chris Manning, 467-479. Stanford: CSLI Publications.

Jesney, Karen (2007) The locus of variation in weighted constraint grammars. Paper given at the Workshop on Variation, Gradience and Frequency in Phonology, Stanford, CA.

Jun, Jongho (2015) Korean n-insertion: a mismatch between data and learning. *Phonology* 32(3):417-458.

Jurafsky, Dan and James H. Martin (2025) *Speech and Language Processing* (draft in progress), web.stanford.edu/~jurafsky/slp3/

Kager, René. 1999, *Optimality theory*. Cambridge: Cambridge University Press.

Kaplan, Aaron (2018) Positional licensing, asymmetric trade-offs and gradient constraints in Harmonic Grammar. *Phonology* 35:247-286.

Kaun, Abigail R. (2004) The typology of rounding harmony. In *Phonetically Based Phonology*, ed. Bruce Hayes, Robert Kirchner and Donca Steriade, 87–116.

Kawahara, Shigeto (2020) A wug-shaped curve in sound symbolism: The case of Japanese Pokémon names. *Phonology* 37:383-418.

Kawahara, Shigeto (2021) Testing MaxEnt with sound symbolism: A stripy wug-shaped curve in Japanese Pokémon names. *Language* 97: e341-e359.

Kiparsky, Paul. 1973. Phonological representations. In Osamu Fujimura, ed. *Three Dimensions in Linguistic Theory*. Tokyo: TEC Co.

Kiparsky, Paul 1993. An OT perspective on phonological variation. Presented at Rutgers Optimality Workshop, New Brunswick, NJ, Oct. 22–24. https://web.stanford.edu/~kiparsky/Papers/nwave94.pdf

Kirchner, Robert. *An effort based approach to consonant lenition*. Xxx place:Routledge, 2013.

Kisseberth, Charles W. 1970. On the functional unity of phonological rules. *Linguistic Inquiry* 1:291–306.

Labov, William (1969) Contraction, deletion, and inherent variability of the English copula. *Language* 45:715-762.

Labov, William (1972) *Sociolinguistic Patterns*. Philadelphia: University of Pennsylvania Press.

Lavoie, Lisa 2001 dissertation xxx

Legendre, Geraldine, Yoshiro Miyata & Paul Smolensky (1990) Harmonic Grammar – A formal multi-level connectionist theory of linguistic well-formedness: An Application. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, 884–891. Mahwah, NJ: Lawrence Erlbaum Associates.

Linzen, Tal, Sofya Kasyanenko, and Maria Gouskova. 2013. Lexical and phonological variation in Russian prepositions. *Phonology* 30(3):453–515.

Lombardi, Linda. 1996. Restrictions on direction of voicing assimilation: an OT account. University of Maryland Working Papers in Linguistics 4, 84-102. [ROA-247.]

Lombardi, Linda (1999) Positional faithfulness and voicing assimilation in Optimality Theory. *NLLT* 17: 267-302.

McCarthy, John. 2002. *A thematic guide to Optimality Theory.* Cambridge: Cambridge University Press.

McCarthy, John. 2008. *Doing Optimality Theory: Applying theory to data*. Oxford: Blackwell.

McCarthy, John and Alan Prince (1995). Faithfulness and reduplicative identity. In Jill Beckman, Suzanne Urbanczyk and Laura W. Dickey (eds.) *University of Massachusetts occasional papers in linguistics 18: Papers in Optimality Theory*. 249–384.

McCollum, Adam G. (2018) Vowel dispersion and Kazakh labial harmony. *Phonology* 35:287-326.

McPherson, Laura. 2013. *A Grammar of Tommo So*. Berlin: De Gruyter.

McPherson, Laura and Bruce Hayes (2016) Relating application frequency to morphological structure: the case of Tommo So vowel harmony. *Phonology* 33:125–167.

Mielke, Jeff (2005) Modeling distinctive feature emergence. *Proceedings of the West Coast Conference on Formal Linguistics*, 24: 281-289.

Moore-Cantwell, Claire (2020) Weight and final vowels in the English stress system. *Phonology* 37:657-695.

Moore-Cantwell, Claire, and Joe Pater (2016) Gradient exceptionality in Maximum Entropy Grammar with lexically specific constraints. *Catalan Journal of Linguistics* 15:53-66.

Pater, Joe (2000) Non-uniformity in English secondary stress: the role of ranked and lexically specific constraints. *Phonology* 17:237-274.

Pater, Joe (2009) Weighted constraints in generative linguistics. *Cognitive Science* 33:999-1035.

Pater, Joe. 2010. Morpheme-Specific Phonology: Constraint Indexation and Inconsistency Resolution. In Steve Parker (ed.). Phonological Argumentation: Essays on Evidence and Motivation, 123-154. London: Equinox.

Prince, Alan S. (2002a). Anything goes. In T. Honma, M. Okazaki, T. Tabata, & S. Tanaka (Eds.) *A new century of phonology and phonological theory* (pp. 66–90). Tokyo: Kaitakusha

Prince, Alan S. (2002b) Arguing optimality. *University of Massachusetts Occasional Papers in Linguistics*: Vol. 28, Article 10.

Prince, Alan & Paul Smolensky (1993/2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Technical report, Rutgers University Center for Cognitive Science. [Published 2004; Oxford: Blackwell]

Riggle, Jason. 2009. Generating contenders. Rutgers Optimality Archive 1044. xxx address

Riggle, Jason. 2010. Sampling Rankings. Rutgers Optimality Archive, http://roa.rutgers.edu/article/view/1105. xxx

Rosenthall, Sam (1989). The phonology of nasal-obstruent sequences. xxx

Rousseau, Pascale and David Sankoff (1978) Advances in variable rule methodology. In David Sankoff, ed., *Linguistic variation: Models and methods*, pp. 57-69.

Ryan, Kevin (2019) *Prosodic Weight: Categories and Continua*. Oxford: Oxford University Press.

Sankoff, David, and William Labov (1979) On the uses of variable rules. *Language in Society* 8: 189-222.

Scholes, Robert (1965) *Phonotactic Grammaticality.* The Hague: Mouton.

Shih, Stephanie (2018) Learning lexical classes from variable phonology. In *Proceedings of the Second Asian Junior Linguists Conference*. Tokyo: International Christian University.

Smith, Brian W. and Joe Pater (2020) French schwa and gradient cumulativity. *Glossa: a journal of general linguistics* 5:24.

Smith, Jxxx. L. (2001). Lexical category and phonological contrast. In *PETL 6: Proceedings of the Workshop on the Lexicon in Phonetics and Phonology* (pp. 61-72). Edmonton: University of Alberta.

Smolensky, Paul (1986) Information processing in dynamical systems: Foundations of Harmony Theory. In James L. McClelland, David E. Rumelhart and the PDP Research Group. *Parallel distributed processing.* Cambridge: MIT Press. 390-431.

Smolensky Paul and Geraldine Legendre, (2006)*The Harmonic Mind*

Storme, Benjamin (forthcoming) Not only size matters: limits to the Law of Three Consonants in French phonology. To appear in *Glossa*.

Walker, James A. (2010). *Variation in Linguistic Systems (1st ed.)*. Routledge. https://doi.org/10.4324/9780203854204

Wilson, Colin (2006) Learning phonology with substantive bias: an experimental and computational investigation of velar palatalization. *Cognitive Science* 30.945–982.

Wilson, Colin and Marieke Obdeyn (2009) Simplifying subsidiary theory: statistical evidence from Arabic, Muna, Shona, and Wargamay. Ms., Johns Hopkins University.
Wolfram, Walt. 1993. Identifying and interpreting variables. In Dennis Preston (ed.), *American Dialect Research*. Amsterdam: Benjamins, 193–221.
Wolfram, Walt, and Ralph W. Fasold (1974) The study of social dialects in American English. Englewood Cliffs, NJ: Prentice Hall.
Zhang, Jie et al. (2011) xxx Taiwanese tone sandhi paper
Zuraw, Kie (2000) Patterned exceptions in phonology. Ph.D. dissertation, UCLA.
Zuraw, Kie (2001) xxx figure out what Claire meant
Zuraw, Kie (2002) Aggressive reduplication. *Phonology* 19: 395-439.
Zuraw, Kie (2010) A model of lexical variation and the grammar with application to Tagalog nasal substitution. *Natural Language & Linguistic Theory*, 28: 417-472.
Zuraw, Kie and Bruce Hayes (2017) Intersecting constraint families: an argument for Harmonic Grammar. *Language* 93:497-548.