

Supplementary materials for Hayes and Kaplan (2023) “Zero-Weighted Constraints in Noisy Harmonic Grammar”

Linguistic Inquiry Squibs and Discussion

1. Materials for Tableau (1)

1.1 Purpose

Show that weights of values 4.95 for PREFERMAJORITY and 4 for PREFERMINORITY derive a .75/.25 probability split.

1.2 Files

1_SpreadsheetForTableau1.xlsx
1_VerifyWeights.txt

1.3 Discussion

In this and other spreadsheets, we construct a **Diagnostic Gaussian**, following the method presented by Kie Zuraw (2000, UCLA dissertation; <https://linguistics.ucla.edu/people/zuraw/dnldpprs/diss.pdf>), and more recently by Flemming (2021).

The mean of the Diagnostic Gaussian is the sum of the base weights of the constraints that prefer the Minority candidate, minus the sum of base weights of the constraints that prefer Majority. Its standard deviation is the square root of the sums of the squared standard deviations of all constraints that prefer one or the other input (this works because no constraint is violated more than once; otherwise one must count each violation separately in the total). The probability that the Majority candidate wins is the probability that the Diagnostic Gaussian will go above zero. The latter value can be looked up using the NORMDIST() function of Excel. (The four arguments of this function are: the critical value of zero, the mean, the standard deviation, and “TRUE” to indicate that the cumulative distribution is required.)

The spreadsheet calculates the probability of the candidates from the constraint weights. The weight of PreferMinority was set at 4, and the weight of PreferMajority was set to ensure that the output probabilities would match the target values .75/.25.

The OTSoft output file, 1_VerifyWeights.txt, simply verifies that the Excel-calculated weights work, using the sampling method (running the grammar for many evaluation times). It was run in OTSoft 2.6 (<https://linguistics.ucla.edu/people/hayes/otsoft/>), using the OTSoft tab of the Excel spreadsheet as the input file. Sampling tends not to be as accurate as other methods and the .75/.25 target is only approximated.

2. Materials for Section 3, Tableau (4), “Failure of harmonic bounding in one version of Noisy Harmonic Grammar”

2.1 Purpose

Show that if perturbed weights are allowed to go below zero, harmonic bounding is subverted; the bounded candidate receives a substantial probability.

2.2 Files

2_SpreadsheetForTableau5.xlsx
2_VerifyWeights.txt

2.3 Discussion

The Excel spreadsheet applies the Diagnostic Gaussian method given above, and the second file verifies the weights in OTSoft.

3. Materials for Section 4, Tableau (5), “Assessing the effect of a zero-weighted constraint aligned with PREFERMAJORITY”

3.1 Purpose

Show that if Clipping is enforced, a zero-weighted constraint can affect probability, penalizing a candidate that violates it.

3.2 Files

3_SpreadsheetForTableau6ClippingEnforced.xlsx
3ResultsForTableau6ClippingEnforced.txt

3.3 Discussion

In this case, we used just OTSoft to calculate the output probabilities, since as far as we know Excel cannot be used to cover cases with clipping. The Excel spreadsheet is there simply to serve as the input file for OTSoft. The answer OTSoft found is given in 3ResultsForTableau6ClippingEnforced.txt.

4. Section 4.1.1: maximum size of effect (Clipping in force, 1 violation of MAJORITY HELPER)

4.1 Purpose

Find out the largest possible size of the effect of a zero-weighted MAJORITYHELPER for tableau (5), which depends on the weight difference between PREFERMAJORITY and PREFERMINORITY. In this case we assume clipping. The answer was found with a brute-force search executed with a small computer program.

4.2 Files

4_Code.pdf

4LargestNegativeEffectOfAZeroWeightedConstraint.xlsx

4.3 Discussion

The search was performed with a simple program written in Basic. Program code is included for verification. The program looks at a range of values for the weight difference between PREFERMAJORITY and PREFERMINORITY (only the difference matters), assuming one violation of zero-weighted MAJORITYHELPER. The program runs a large number of evaluation times for each weight-difference value, and prints out the probability of the Majority and Minority candidates, so one can locate the approximate point where the effect of MAJORITYHELPER is maximized. The output of the program is plotted in the spreadsheet.

5. Section 4.1.1: maximum size of effect (many violations of MAJORITY HELPER)

Here there is no software and we work simply with reasoning, as follows.

The scenario we put forth as the probable maximum is this. PREFERMINORITY is much stronger than PREFERMAJORITY, but the number of violations of MAJORITYHELPER is huge, so huge as to almost always override the effect of PREFERMINORITY where they conflict. Then:

1. For Input1, the Minority candidate will almost always win, per the weights just given.
2. Next consider Input2, in which the Minority candidate also violates MAJORITYHELPER. If MAJORITYHELPER's weight is *not* clipped, it will almost always determine the outcome: the enormous right tail of its distribution means that the Majority candidate will almost always win.
3. If MAJORITYHELPER's weight *is* clipped, it has no effect, no matter how many violations it has, since zero times any number is zero. In this case, the decision will almost certainly favor the Minority candidate, since PREFERMINORITY is much stronger than PREFERMAJORITY.
4. Hence, the decision comes down, in the limit, to a coin flip for whether MAJORITYHELPER gets clipped or not, and is therefore 50/50 in the limit.
5. Thus we compare slightly-above-0 probability of the Majority candidate for Input1, and the almost 0.5 probability for Majority candidate for Input2, a difference of almost 0.5.
6. In intuitive terms: MAJORITYHELPER can have enough violations to virtually always determine the outcome when not turned off, but it is turned off half the time.

6. Materials for Section 4.2 (Tableau (5)) — Clipping not enforced

6.1 Purpose

Show that if perturbed weights are allowed to go below zero, MAJORITYHELPER can actually lower the probability of the Majority candidate.

6.2 Files

6SpreadsheetForTableau6ClippingNOTEnforced.xlsx
6_VerifyWeights.txt

6.3 Discussion

The Excel spreadsheet applies the Diagnostic Gaussian method given above, and the second file verifies the weights in OTSoft.

7. Materials for Section 4.2 — maximum size of effect (Clipping not enforced, 1 violation of MAJORITY HELPER)

7.1 Purpose

Assuming Clipping is not in force, find out the maximum that a single violation of MAJORITYHELPER can hurt the Majority candidate, following tableau (5). This depends on the weight difference between PREFERMAJORITY and PREFERMINORITY.

7.2 File

7MaximumThatOneViolationOfMajorityHelperCanHurtMajorityCandidate.xlsx

7.3 Discussion

This can be done with the same basic spreadsheet as for section 3 above. In this case, we used the Excel Solver, manipulating the weight of PREFERMAJORITY, to find the value at which Majority Helper maximally harms the Majority candidate. This turns out to be about 5.56, where the degree of harm is 0.049.

The second tab of the spreadsheet plots the probability of the Majority candidate for Input1 and Input2 for a range of values of the weight of PREFERMAJORITY, keeping the weight of PREFERMINORITY constant at 4. The graph also plots the difference between the two probabilities. We obtain an undulating curve, representing the influence of MAJORITY HELPER across the spectrum.

8. Materials for Section 4.2 — maximum size of effect (Clipping not enforced, many violations of MAJORITY HELPER)

Here there is no software and we just employ reasoning, as follows.

The scenario is exactly the same as in §5 above, and the following mostly repeats the wording of §5.

1. For Input1, the Minority candidate will almost always win, as shown before.
2. Next consider Input2, in which the Minority candidate also violates MAJORITYHELPER. When MAJORITYHELPER's weight is positive, it will almost always determine the

outcome: the enormous right tail of its distribution means that the Majority candidate will almost always win.

3. When MAJORITYHELPER's weight is negative, it helps the Minority candidate. But this candidate would win anyway, since by hypothesis PREFERMINORITY has a much higher weight than PREFERMAJORITY.
4. Hence, the decision comes down, in the limit, to a coin flip for whether MAJORITYHELPER is above zero or not, and is therefore 50/50 in the limit.
5. Thus we compare slightly-above-0 probability of Majority candidate for Input1, and almost 0.5 probability for Majority candidate for Input2, a difference of almost 0.5.
6. In brief, intuitive terms: MAJORITYHELPER can have enough violations to have an extremely powerful effect, but it only favors the Majority candidate half the time.

9. End of section 4.2: MAJORITY HELPER can hurt Majority candidate even when it has a modest positive weight

9.1 Purpose

Assuming Clipping is not in force, explore the positive weight region for MAJORITYHELPER, establishing values for which MAJORITYHELPER hurts the Majority candidate. PREFERMAJORITY and PREFERMINORITY are set to yield 75/25 probability for Input1; the graph plots the effects of MAJORITYHELPER as its weight is varied. We find that MAJORITYHELPER is a majority-hurter when its weight is less than 0.214.

9.2 File

9MajorityHelperAlsoHurtsMajorityCandidateWhenItHasASmallPositiveValue.xlsx