OT-Help User Guide

Michael Becker and Joe Pater

University of Massachusetts, Amherst

1. Introduction

OT-Help (Becker, Pater & Potts 2007) is a free open-source Java-based program that aids research in Optimality Theory (OT, Prince and Smolensky 1993/2004) and its cousin, Harmonic Grammar (HG, Smolensky and Legendre 2006). OT-Help will find a constraint ranking or weighting consistent with the data provided by the user, if one exists. The solution can be displayed either in a standard tableau format, or as a comparative tableau (Prince 2002a). OT-Help will also find the set of possible languages in both OT and HG, given the user's set of constraints. Typology can also be explored interactively by selecting different sets of optima.

In the following, we provide a guide to the use of OT-Help, as well as a brief introduction to the theory underlying it, especially with respect to HG, which is less well known than OT.

In both OT and in the version of HG we are working with, a language is a set of input-output pairs that are optimal under some ranking or weighting of a set of constraints. As the number of constraints, inputs, and candidates increases, it becomes increasingly difficult to find by hand a ranking for an intended set of optima, and especially to find by hand the set of possible languages. The need for an automated means of finding rankings and calculating typologies was recognized early on in the development of OT, and been met by OT-Soft (Hayes, Tesar and Zuraw 2003) and Praat (Boersma and Weenink 2007). With HG's weighted constraints, these problems are multiplied. OT-Help is the first tool that allows a user to quickly calculate the set of languages generated by a given set of weighted constraints.

2. Ranking conditions and weighting conditions

In (1), we present a tableau for an input /kata/ with two output candidates: the intended optimum, or 'winner' (káta) has a trochee, or left-headed foot, and the sub-optimal 'loser'

(katá) has an iambic, right-headed foot. There are two constraints: IAMB, which demands that all feet be right-headed, and TROCHEE, which demands that all feet be left-headed.

(1)

/kata/		IAMB	TROCHEE
→	a. káta	*	
	b. katá		*

We see that the constraint TROCHEE assigns no violation marks to the winner and one violation mark to the loser, or in other words, TROCHEE prefers the winner to the loser. The opposite is true of IAMB, which prefers the loser. The same information can be represented in a comparative tableau (Prince 2002a), where W means "winner-preferring" and L means "loser-preferring".

(2)

/kata/	IAMB	TROCHEE
→ káta ~ katá	L	W

A necessary property of a ranking that produces the intended result is that every loser-preferring constraint be dominated by some winner-preferring constraint (Prince and Smolensky 1993/2004, Tesar and Smolensky 1998, Prince 2002a). In our case, IAMB must be dominated by TROCHEE, so the ranking we want is TROCHEE >> IAMB.

Prince and Smolensky (1993/2004: 236) consider (and reject) an alternative formulation of OT that replaces constraint ranking with weighting, as in Harmonic Grammar (HG; Smolensky and Legendre 2006). For a review of subsequent research pursuing this alternative, and a comparison with OT, see Pater, Bhatt and Potts (2007a). We use the translation of OT to HG introduced in Legendre, Sorace and Smolensky (2006). Violation marks are translated into negative integers, and a candidate's *harmony* is calculated by multiplying its violation score on each constraint by that constraint's weight, and summing these products. The candidate with the maximal harmony is optimal. Following Prince (2002b) and Pater et al. (2007a), we restrict weights to positive reals. OT-Help also allows users to input positive constraint satisfaction scores, and to fix the minimal values of constraint weights at zero (see Keller 2006), or any other non-negative value.

Returning to our example, the harmony of *káta* is the weight of IAMB times -1, plus the weight of TROCHEE times zero, which equals the negative weight of IAMB (3). The similar calculation for *katá* is shown in (4).

(3) $H(\text{káta}) = w(\text{IAMB})\bullet\text{-}1 + w(\text{TROCHEE})\bullet 0 = \text{-}1\bullet w(\text{IAMB})$

(4) $H(\text{katá}) = w(\text{IAMB})\bullet 0 + w(\text{TROCHEE})\bullet\text{-}1 = \text{-}1\bullet w(\text{TROCHEE})$

(5)

/kata/		IAMB	TROCHEE
→	a. káta	-1	
	b. katá		-1

Since *káta* is the winner, its harmony must be higher than the harmony of *katá* (6a), which entails that the weight of IAMB must be smaller than the weight of TROCHEE (6c):

(6) a. H(káta) > H(katá)
b. -1•w(IAMB) > -1•w(TROCHEE)
c. w(IAMB) < w(TROCHEE)

Statements of the form in (6c) are HG's weighting conditions, which are parallel to OT's Elementary Ranking Conditions (Prince 2002a). We extend the notion of comparative tableau to HG, as in (7). In an HG comparative tableau, the numerical difference between the winner and the loser is indicated by a positive score for a winner-preferring constraint and a negative score for a loser-preferring constraint. Given a weighting that correctly chooses the optima, the sum of the weighted differences will be positive.

(7)

/kata/	IAMB	TROCHEE
→ káta ~ katá	-1	+1

Of the infinitely many sets of weights that achieve the intended result, we chose to set the weight of IAMB at 1 and the weight of TROCHEE at 2. The outcome is shown below, where the Harmony of each candidate is displayed in the rightmost column:

(8)

	Weights:	1	2	
/kata/		IAMB	TROCHEE	*H*
→	a. káta	-1		-1
	b. katá		-1	-2

The hamony of *káta* is -1, which is higher than the harmony of *katá*, so *káta* is chosen as optimal. Note that if the winner-loser difference scores in (7) are multiplied by the weights in (8) and then summed, the result is indeed positive (+1).

To illustrate a somewhat more complicated case, we add another constraint, and another input with two candidates.

(9)

/kata/		IAMB	TROCHEE	*STRESS/HI
→	a. káta	-1		
	b. katá		-1	
/pika/				
	a. píka	-1		-1
→	b. piká		-1	

*STRESS/HI assigns a violation to a stressed high vowel, such as first vowel in *píka*. The question is whether there is a weighting that will pick a trochee as optimal for /kata/, but an iamb as optimal for /pika/, as in the indicated optima in (9).

To make *káta* win over *katá*, the weight of IAMB must be smaller than the weight of TROCHEE, as we've just seen. Now we can add the weighting conditions for the next tableau, calculated in (10-12).

(10) H(píka) = w(IAMB)•-1 + w(TROCHEE)•0 + w(*STRESS/HI)•-1
= -w(IAMB) -w(*STRESS/HI)

(11) H(piká) = w(IAMB)•0 + w(TROCHEE)•-1 + w(*STRESS/HI)•0
= -w(TROCHEE)

(12) H(piká) > H(píka)
-w(TROCHEE) > -w(IAMB) -w(*STRESS/HI)
w(TROCHEE) < w(IAMB) + w(*STRESS/HI)

Combining the weighting conditions from (6c) and (12), we arrive at two inequalities that need to be satisfied:

(13) w(IAMB) < w(TROCHEE)
w(TROCHEE) < w(IAMB) + w(*STRESS/HI)

Can we assign weights that will satisfy both? While the answer is perhaps less obvious than in the last case, one weighting that works is w(IAMB) = 1, w(TROCHEE) = 2, w(*STRESS/HI) = 2. The tableau below shows that these weights assign the highest harmony scores to the intended winners:

(14)

Weights:		1	2	2	
/kata/		IAMB	TROCHEE	*STRESS/HI	*H*
→	a. káta	-1			-1
	b. katá		-1		-2
/pika/					
	a. píka	-1		-1	-3
→	b. piká		-1		-2

We can also examine the possibility that in some language, *píka* would win over *piká*, and *katá* over *káta*. Is there a weighting of our constraints that would produce this result? The required weighting conditions are below:

(15) H(káta) < H(katá)
-1•w(IAMB) > -1•w(TROCHEE)
w(IAMB) < w(TROCHEE)

(16) H(piká) < H(píka)
-w(TROCHEE) < -w(IAMB) -w(*STRESS/HI)
w(TROCHEE) > w(IAMB) + w(*STRESS/HI)

Together, we get:

(17) w(IAMB) > w(TROCHEE)
w(TROCHEE) > w(IAMB) + w(*STRESS/HI)

There is no set of non-negative weights we could assign to these three constraints that would satisfy the two requirements above, so we can conclude that HG predicts the impossibility of the language above.

For these small problems, finding correct constraint weightings is not much more difficult than finding rankings. For larger problems, however, finding out whether a set of candidates can be jointly optimal can be considerably more difficult in HG than in OT. To address this difficulty, Pater, Potts and Bhatt (2007) introduce an application of Linear Programming that returns a weighting if one exists, and indicates when no weighting of the constraints will yield the intended optima. A web-based implementation of this application is available as Potts, Becker, Bhatt and Pater (2007). OT-Help includes this implementation, and extends it to the calculation of HG typologies.

3. Using OT-Help: The data file

OT-Help is designed to read files in an extension of Hayes et al.'s (2003) OT-Soft format. These files are easiest to create in a spreadsheet program, such as the free OpenOffice "Calc" or Microsoft Excel. To submit a file to OT-Help, it must be saved as a text file (tab separated), which can then be dropped into the OT-Help window. Files that are not plain-text OT-Soft files will be rejected. The text is read as HTML, so special characters should be entered as they would be in a regular HTML file. Unicode files will be read, but special characters will not display correctly.

The table below shows what the tableaux in (14) looks like in OT-Soft format:

(18)

			Iamb	Trochee	*Stress/Hi
			Iamb	Trochee	*Stress/Hi
kata	káta	1	1		
	katá			1	
pika	píka		1		1
	piká	1		1	

The first column is the input column. Each new input defines the beginning of a new tableau. The next column is the candidate column. In the example above, we see two tableaux, each with one input and two candidates. The third column is the winner marking column. In this column, any non-zero number will serve to mark an intended optimum. The current version of OT-Help only allows a single optimum per tableau. Note that under the definitions of optimality that we are working with, the optimum must be evaluated as better than all other candidates; a tie is not sufficient. Therefore, if an intended optimum has the same violation marks as another candidate, it cannot be made optimal, and OT-Help will fail to find a solution in either OT or HG.

The first two rows are constraint names. You will see that most OT-Help screens only display the names from the second line of your file, but following OT-Soft format, both lines are required. Finally, the space below the first two lines and to the right of the first three columns contains violation profiles. Again following OT-Soft format, these are inserted as positive integers, though they display in OT-Help HG tableaux as negative integers. If negative integers are entered into the OT-Soft file, they will display as positive integers in the OT-Help display, and be treated as satisfaction scores (see Legendre et al.'s 2006 discussion of 'positive HG'). Zeros may be left unspecified.

After the first portion of the input file, which follows the OT-Soft format exactly and is required by OT-Help, optional parameters may be added. This is marked by the addition of a new line, after the last tableau, that reads "[end of tableaux]".

Among the options that are already implemented in the current version of OT-Help is the "[minimal weight]" directive, which allows the specification of minimal HG weights. In the absence of user specification, a default value of 1 is assumed. To specify a default minimal value for all constraints, enter any number in the second column of the "[minimal weight]" row. To specify a minimal weight for some specific constraint, enter a number under its column.

In the example below, OT-Help will set 0.5 as the minimal weight for all constraints, and 3 as the minimal weight for TROCHEE.

(19)

			Iamb	Trochee	*Stress/Hi
			Iamb	Trochee	*Stress/Hi
kata	káta	1	1		
	katá			1	
pika	píka		1		1
	piká	1		1	
[end of tableaux]					
[minimal weight]	0.5			3	

Note that you may specify a minimal weight of zero, and this will be different from not specifying a minimum, since the default value in OT-Help is 1. This is one case where leaving a blank is different from specifying a zero.

The minimal weight values are irrelevant to OT calculations, and are ignored. In future versions of OT-Help, we expect to add other user-specifiable options, such as biases for particular constraints to be ranked or weighted higher or lower than other constraints.

4. OT-Help navigation

OT-Help opens with a welcome screen, onto which you are invited to drop your input file. If a valid file is received, and each tableau in the file specifies a unique winner, three options are offered:

- Find all solvable languages (HG & OT)
- HG Solution
- OT Solution

If some tableaux in your file specify more than one winner, only the first option will be offered. We now cover the three options in turn.

4.1 Find all solvable languages (HG & OT)

This option will find all the solvable *languages* in your file. Recall that in OT, a language means a set of winners that can be made jointly optimal. In a solvable language, there is a weighting (and possibly also a ranking) of the constraints that picks out the winner from each tableau. OT-Help comes with two solvers:

- A solver for Harmonic Grammar, written by Christopher Potts, which uses the linear programming translation from Pater et al. (2007b).
- A solver for Optimality Theory, written by Michael Becker, which uses the Recursive Constraint Demotion algorithm (Tesar and Smolensky 1998, Prince 2002a).

When given a set of intended optima, a solver returns a verdict (solvable or not solvable), and if a solution is found, it is returned as well, in the form of a weighting or ranking of the constraints.

In calculating typologies, OT-Help follows a method introduced in Hayes et al.'s (2003) OT-Soft. It starts by looking just at the first tableau of the input file. Each candidate is tried out as a winner, and OT-Help attempts to find a weighting of the constraints that chooses that candidate as a winner. If no candidate can be winner, the search for solvable languages ends, since no matter what other tableaux specify, no winner can be chosen from the current tableau.

In the more likely event that a subset of the candidates in the first tableau are possible winners, each one of them is tried out with each of the candidates of the second tableau, looking for weightings that pick out the winners from each tableau. This continues with the remaining tableaux, until OT-Help finds the list of sets of winners that have a weighting.

Once all the languages that are solvable in HG are found, each one is passed on to the RCD module, to find a ranking of the constraints that picks out the same winners, if one exists. We can solve for HG first because any set of winners that can be made optimal in OT

can also be made optimal in HG.[1] Following this method allows us to efficiently compare the results in the two theories. In future versions of OT-Help, we expect to make available the possibility of calculating just the OT typology, which could be useful for large problems.

When done, OT-Help will display all the languages it found, as shown in (20), starting with ones that are solvable in both HG and OT, and then displaying languages that are only solvable in HG. The list of languages is saved and will not be recomputed until OT-Help is closed or a new input file is accepted.

(20)

Your input file contains 2 tableaux and a total of 4 candidates. These can potentially give rise to 4 different languages.

An HG solution was found for 3 languages. An OT solution was found for all of them.

Languages that have an HG solution and an OT solution:

1. [kata -> káta, pika -> píka] HG Solution OT Solution
2. [kata -> káta, pika -> piká] HG Solution OT Solution
3. [kata -> katá, pika -> piká] HG Solution OT Solution

Make .Collection file (Praat)

Back to home page of the current file

At the bottom of the page is a link "Make .Collection file (Praat)". When this link is clicked, OT-Help will create a new text file in the same directory as the input file, which can be read by Praat (Boersma and Weenink 2007). It contains a grammar file that includes all of the tableaux, and data files for each of the sets of winners that could be rendered jointly optimal by some weighting of the HG constraints. The resulting file can be opened directly in Praat. One can then test the various learners implemented in Praat on their abilities to find a correct ranking or weighting for each of the languages.

We have used the current version of OT-Help to find the predicted typologies for very large tableaux sets (see the discussion of the Kager 2006 typology in Pater et al. 2007b). It does, however, seem to have limits on the size of problems that it can process, which we hope to overcome in future versions.

4.2 HG Solution

The HG Solution page can be reached either from an input file's home page, or from the list of solvable languages. This page starts with the verdict, as in (21), stating whether a solution was found by the HG solver or the OT solver. To make the verdict more salient, the page's background color will be a light blue if a solution was found, and an alarming pink if no solution could be found.

[1] It is theoretically possible that the linear programming solver we use in our implementation could misclassify sets of winners as infeasible, due to floating point errors, for example, but we have yet to see an example of such an error.

(21)

RCD status: Solved.

Click on a loser to make it into a winner.

LP status: solved

Min constraint weight was set to 1.0

Weights	1.0	3.0	1.0	
Input: kata	Iamb	Trochee	*StressHi	
--> káta	-1.0			Weighted total: -1.0
katá		-1.0		Weighted total: -3.0

Weights	1.0	3.0	1.0	
Input: pika	Iamb	Trochee	*StressHi	
--> píka	-1.0		-1.0	Weighted total: -2.0
piká		-1.0		Weighted total: -3.0

Switch to comparative view

Back to list of languages

Back to home page of the current file

To process a new OTSoft file, drop it in this window.

Next, if the HG solver returned a successful verdict, the weights that were found for the constraints are shown in each tableau. Note that if a tableau has any losers in it, you can click on a loser to make it into a winner. This will refresh the page, passing the new language to the HG solver. If an infeasible verdict was returned by the solver, the weights are set to their minimal default values.

The HG solver finds the minimal weighting, that is, the set of weights whose summed total is as small as possible. This minimization is bounded in two ways. The first is by the minimal constraint weights discussed above. The second is by a condition that difference between the winner's harmony and any loser must be at least 1. See Pater et al. (2007b) for further discussion.

In addition to the standard view, which is displayed by default, there is a comparative view, available from a link at the bottom of the page. Once clicked, winners and losers and displayed in pairs, and instead of raw violation scores, difference in violation scores are displayed (see also Goldwater & Johnson 2003). The last column displays the total weighted difference between winners and losers.

This view is useful, for instance, for easily spotting harmonically bounded candidates, which will have nothing but zeroes and positive numbers (if the harmonically bounded candidate is a loser) or zeroes and negative numbers (if the harmonically bounded candidate is a winner). The comparative view is also useful for spotting sets of winner-loser pairs with identical vectors of differences in violation scores. Since each winner-loser pair in such a set

contributes the same information to learning, the analysis would be unchanged if only one pair in the set was kept and the rest were discarded.

4.3 OT Solution

The OT Solution page can be reached either from an input file's home page, or from the list of solvable languages. This page starts with the verdict (22), stating whether the RCD-based solver found a ranking. This verdict is also reflected in the page's background color, with blue for solvable systems and pink for insolvable ones.

(22)

RCD status: Solved

The ranking found: Trochee >> Iamb, *StressHi

Click on a loser to make it into a winner.

input	winner ~ loser	Trochee	Iamb	*StressHi
kata	káta ~ katá	W	L	
pika	píka ~ piká	W	L	L

Switch to standard view

Back to home page of the current file

If a ranking was found, it is reported. Otherwise, constraints that could not be ranked are listed. This list can help in identifying what inconsistent data prevented the solver from finding a ranking.

In standard view, the tableaux from the input files will be presented as entered, with asterisks for violation marks. Any losers will be clickable to make them into winners. In comparative view, winners and losers are displayed in pairs, and differences in violation marks are noted with W (winner-preferring), L (loser-preferring), or blank, following the convention in Prince (2002a). Such winner-loser pairs and their vector of W's, L's and blanks are also known as ERC's, for Elementary Ranking Conditions (Prince 2002a).

In unsolvable systems, ERC's that were left unaccounted for, i.e. that could not be used to make a ranking argument are shown in red. This is useful for finding any sources of inconsistent ranking arguments present in the data.

The comparative view also notes ERC's that have at least one L but no W's, pointing out that with such an ERC in the set of tableaux, no ranking could be found. Conversely, ERC's that don't have any L in them are removed from the list of ERC's for redundancy. Finally, only one from any set of identical ERC's will be kept in the list of ERC's, to avoid redundancy.

5. Conclusions

We hope that OT-Help will serve as a useful tool for the study of ranked and weighted constraint interaction. We also hope that its implementation as a set of java classes will facilitate the use of its components in future applications. We welcome correspondence from users with suggestions on how it could be improved in future versions, and also from developers who would like to expand on its capabilities themselves.

References

Boersma, Paul and David Weenink. 2007. *Praat: doing phonetics by computer*. Software version 4.6.09. Available at www.praat.org.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw. 2003. OTSoft 2.1. software package, http://www.linguistics.ucla.edu/people/hayes/otsoft/

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.

Kager, R.W.J. 2006. "Rhythmic licensing: An extended typology". In *Proceedings of the 3rd International Conference on Phonology,* 5-31. Seoul: The Phonology-Morphology Circle of Korea.

Keller, F. 2006. Linear Optimality Theory as a Model of Gradience in Grammar. In *Gradience in Grammar: Generative Perspectives*, ed. Gisbert Fanselow, Caroline Féry, Ralph Vogel, and Matthias Schlesewsky. Oxford University Press.

Legendre, Géraldine, Antonella Sorace, and Paul Smolensky. 2006. The Optimality Theory–Harmonic Grammar connection. In Smolensky and Legendre (2006), 903–966.

Pater, Joe, Rajesh Bhatt and Christopher Potts. 2007a. Linguistic Optimization. Ms, University of Massachusetts, Amherst.

Pater, Joe, Christopher Potts, and Rajesh Bhatt. 2007b. Harmonic Grammar with Linear Programming. Ms, University of Massachusetts, Amherst. ROA-872.

Potts, Christopher, Michael Becker, Rajesh Bhatt and Joe Pater. 2007. HaLP: Harmonic grammar with linear programming, version 2. Software available online at http://web.linguist.umass.edu/~halp/.

Prince, Alan. 2002a. Arguing Optimality. In *University of Massachusetts Occasional Papers in Linguistics*: *Papers in Optimality Theory II*, ed. Angela Carpenter, Andries Coetzee, Paul de Lacy, 269-304. Amherst, MA: GLSA. [ROA-562].

Prince, Alan. 2002b. Anything Goes. In New century of phonology and phonological theory, ed. Takeru Honma, Masao Okazaki, Toshiyuki Tabata, and Shin ichi Tanaka, 66–90. Tokyo: Kaitakusha. ROA-536.

Prince, Alan, and Paul Smolensky. 1993/2004. *Optimality Theory: Constraint interaction in generative grammar*. Ms, Rutgers University and University of Colorado at Boulder, 1993. Revised version published by Blackwell, 2004. [ROA-537].

Smolensky, Paul, and Geraldine Legendre. 2006. *The harmonic mind: From neural computation to Optimality-Theoretic grammar*. Cambridge, MA: MIT Press.

Tesar, Bruce and Paul Smolensky. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29: 229-268.

Department of Linguistics
South College
University of Massachusetts
Amherst, MA 01003

{michael,pater}@linguist.umass.edu