Manual for Maxent Grammar Tool

Bruce Hayes
UCLA
November 2009

1. Introduction: what's it for?

This software computes weights for constraint-based **maxent grammars**. It is meant to be a useful tool for linguists.

The key idea is that you make up a grammar consisting of constraints, then train it, using the program to match a corpus of data. The goal might be to model real-life language learning, or it might simply to make a grammar that's more accurate than one you could produce by hand.

To accomplish this, you need a mathematical expression of the grammar that (a) lets it make quantitative predictions; (b) reliably (provably) yields the optimum grammar compatible with the data.[1] Maxent is, as far as I know, currently the only mathematical grammar framework for constraint-based theories that satisfy these two criteria.

2. What it does

Using an input file, you feed the program:

- Underlying representations
- Rival surface representations for each underlying representation
- Specification of winners — if multiple winners, their frequencies
- A list of constraints
- The number of violations of each constraint for each candidate.

The program computes, and writes to an output file:

- a set of maxent weights for a maxent grammar
- the predicted probabilities that the grammar, when using these weights, assigns to each candidate

3. Where to learn about maxent

- Goldwater, Sharon and Mark Johnson (2003) Learning OT constraint rankings using a Maximum Entropy model. *Proceedings of the Workshop on Variation within Optimality*

[1] By "optimum" is meant: assigns the maximum possible predicted probability to the observed data, within the limits imposed by the constraint system. This is a standard criterion in computer science.

Theory, Stockholm University, 2003. Download from http://homepages.inf.ed.ac.uk/sgwater/papers/OTvar03.pdf.

An explanation for the layman on how the weights get found:

- Hayes, Bruce and Colin Wilson (2008) "A maximum entropy model of phonotactics and phonotactic learning," *Linguistic Inquiry* 39: 379-440. Download from http://www.linguistics.ucla.edu/people/hayes/Phonotactics/HayesWilsonMaximumEntropyPhonotacticsAugust2007.pdf

Further explanation, with a nice application of the method in linguistics:

- Wilson, Colin (2006) Learning phonology with substantive bias: an experimental and computational study of velar palatalization. *Cognitive Science* 30.5:945-982. Download from http://web.jhu.edu/cogsci/people/faculty/Wilson/papers/VelarPalCogSciWilson.pdf

4. How to run the program

4.1 Java

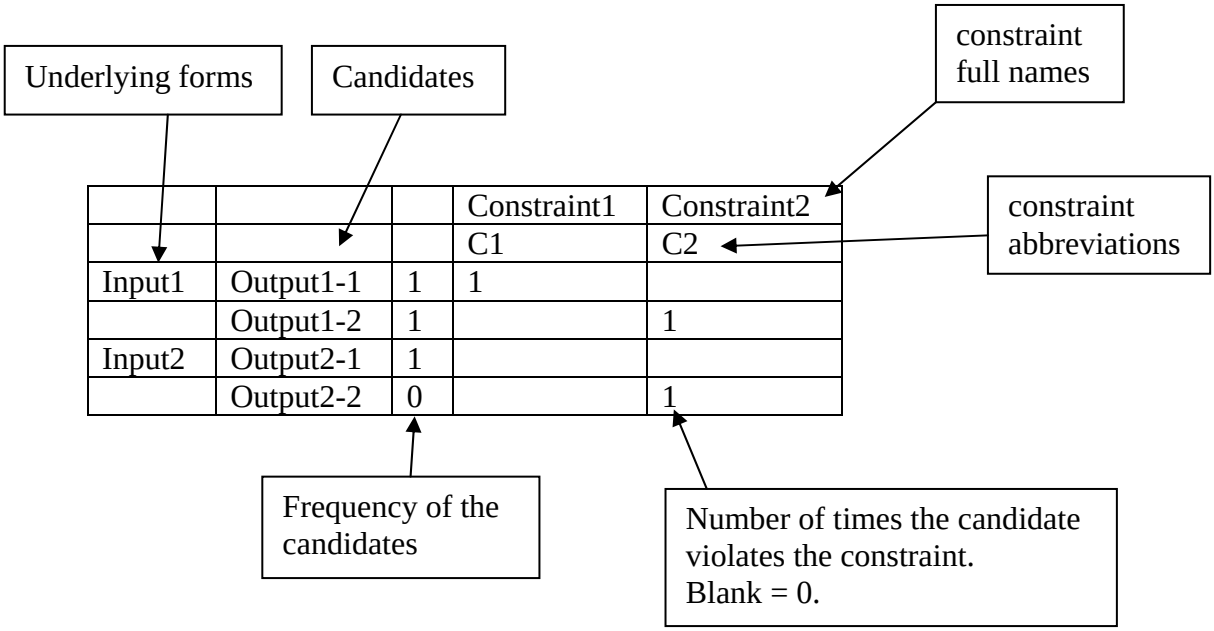
You need Java on your computer. Download from http://www.java.com/en/download/manual.jsp and install.

Java works on various operating systems. For the case of the Macintosh system, you are advised to consult the following page for advice.

http://www.metaphoriclabs.com/articles/installing-java-6-on-mac-os-x/

4.2 Input files

Make a file in the required format, which is the same format as in the "OTSoft" constraint-ranking package (http://www.linguistics.ucla.edu/people/hayes/otsoft/). You can find a template in **SampleDataFile.txt**, included in this software package. The format is ordinary text, tab-delimited. It looks like this:

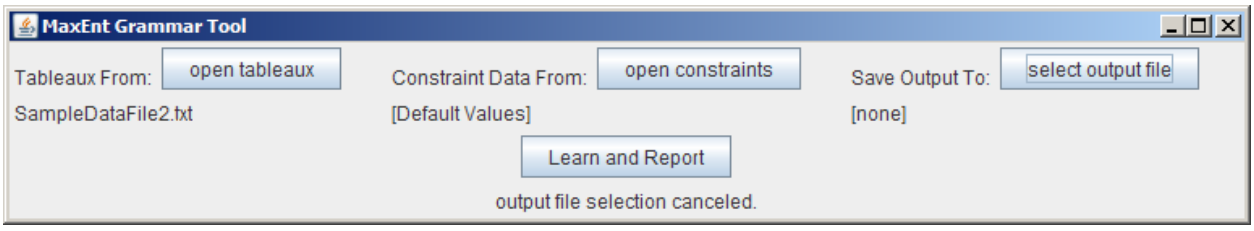


			Constraint1	Constraint2
			C1	C2
Input1	Output1-1	1	1	
	Output1-2	1		1
Input2	Output2-1	1		
	Output2-2	0		1

The text material can be anything you like, though I suggest avoiding exotic characters. The frequencies can be any non-negative number, and the violation counts can be any non-negative whole number. You can use a spreadsheet program to make the file, picking the "Save as tabbed text" option.

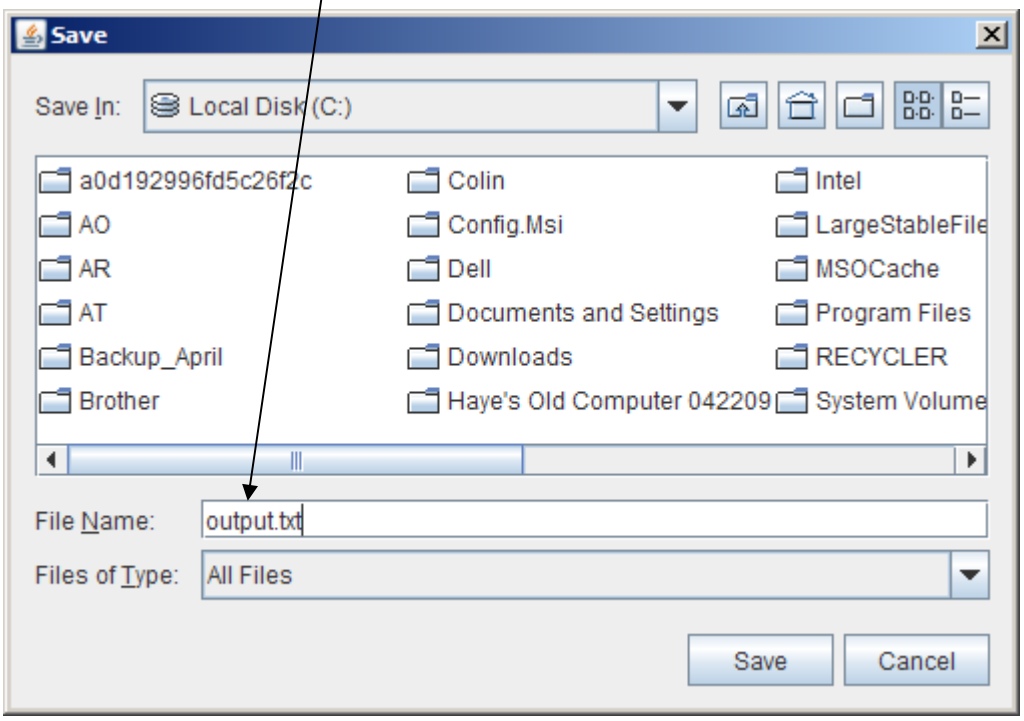
4.3 Launching

Click on **MEGrammarTool.jar**. If your Java is working ok, it should launch and give you this interface:



- Click on **open tableaux**. A "find file" interface will pop up. Navigate through your folders until you reach your input file. Double click.
- Ignore **open constraints** for now (see section 6 below for what this is).

- Click on **select output file** and navigate to the folder where you want your output file. Type in a file name here and then Save.



- This will bring you back to the main interface. Click on **learn and report**.
- If the program is working properly, the **learn and report** button will darken. Otherwise, there's an error; you should check your input file for proper formatting.
- When the program tells you "learning results written successfully", it's done.

4.4 Reading your output file

It's probably easiest to read the output file (plain text format, tab-delimited) using a spreadsheet program such as Excel. Here is a the output file from the input above, which I have prettied up with colors and boxes for clarity.

Input: Input1		C1	C2
Output1-1	1	1	0
Output1-2	1	0	1
Input: Input2		C1	C2
Output2-1	1	0	0
Output2-2	0	0	1

|weights| after optimization:	
Constraint1 (mu=0.0, sigma^2=100000.0)	8.041368461
Constraint2 (mu=0.0, sigma^2=100000.0)	8.041690116

Input:	Candidate:	Observed:	Predicted:
Input1	Output1-1	1	0.500080414
Input1	Output1-2	1	0.499919586
Input2	Output2-1	1	0.999678339
Input2	Output2-2	0	3.22E-04

Part 1: repeats your original input file.
Part 2: the constraints with their calculated weights
Part 3: matchup: how well did the grammar match the original training data? In this case, you can see, it matches the input proportions within about four decimal places.

5. Testing new forms

You can test new forms by including them in the file with all-zero frequencies. So for instance look at the following input file.

Input: Input1		C1	C2
Output1-1	1	1	0
Output1-2	1	0	1
Input: Input2		C1	C2
Output2-1	2	0	0
Output2-2	1	0	1
Input: Input3		C1	C2
Output3-1	0	2	0
Output3-2	0	0	1

This file will cause the system to train the weights based on the first two inputs, then make predictions for the third. The output file tells us this:

Input:	Candidate:	Observed:	Predicted:
Input1	Output1-1	1.0	0.50001
Input1	Output1-2	1.0	0.49999
Input2	Output2-1	2.0	0.66665
Input2	Output2-2	1.0	0.33334
Input3	Output3-1	0.0	0.33335
Input3	Output3-2	0.0	0.66664

6. Gaussian prior (advanced)

You can set a Gaussian prior (see references above) on the learning of the weights, specifying mu and sigma for each constraint. Intuitively, mu is the "preferred" value of the constraint, sigma, when small, tends to force the constraint to be close to mu. Sample file format (tab delimited):

```
C1	-1	10000000
C2	-1	10000000
```

where the first number is mu and second is sigma.

There is a slight glitch in this part of the code: you must enter mu preceded by a negative sign; positive mu values have no effect. However, in output files the mu value is reported without the minus sign.

7. Tech support

This is no tech support for this program. However, you might conceivably be able to get help by emailing Bruce Hayes at bhayes@humnet.ucla.edu. I would also like to hear if there are bugs.

8. Source code

Please contact Bruce Hayes at bhayes@humnet.ucla.edu to request the source code.

9. Credits

Maxent Grammar Tool originated in software prepared by Colin Wilson for purposes of writing Wilson (2006), cited above. The interface and user-friendliness improvements was carried out by Ben George under a grant from the UCLA Academic Senate Council on Research to Bruce Hayes.