Computing assignment: Maximum Entropy grammar

due Thursday, 12 Oct. 2015

Overview and goals

- Develop an analysis of variable data.
- Learn how to use the MaxEnt grammar tool to fit weights to data.
- Play with custom μ and σ values for different constraints.

Step-by-step instructions

Download and test the MaxEnt Grammar Tool

1. Follow the instructions at http://www.linguistics.ucla.edu/people/hayes/MaxentGrammarTool/
2. Read the manual (it's short) and step through running the sample files with it.

Develop an analysis of Shona

3. Take a look at these data, from Uffmann 2007.

Shona has 5 vowels: /a,e,i,o,u/, and the following consonant inventory (Uffmann, p. 46), FYI:

	labials	*alveolars*	*labio-alveolars*	*post-alveolars*	*velars (& 1 glottal)*
stops	p b̤	t d̤			k g̈
implosives	ɓ	ɗ			
affricates	pf bv̤	ts dz̤	tɸ͡s dβ͡z̤	tʃ dʒ̈	
nasals	m m̤	n n̤		ɲ ɲ̈	ŋ ŋ̈
prenasalized stops	mb mb̤	nd nd̤		ndʒ	ŋg
fricatives	f v̤	s z̤	ɸ͡s β͡z̤	ʃ ʒ̈	ɦ̤
prenasalized fricatives	mv	nz	nβ͡z		
liquids		r r̤			
glides	w ʋ			j	

Shona requires every consonant to be following by a vowel (or sometimes [w]), leading to lots of epenthesis. Uffmann analyzes epenthetic vowel quality as predictable from other factors. Here are the rates that he found. Categories are grouped together (/i,e,a,o/), if there was no difference between the sub-categories.

Vowels inserted C__#

preceding V	preceding C	# of i inserted	# u	# e	# o	# a	total	example
i	labial	40	13	0	4	4	61	tim**u** ‘team’
e,a,o,u	labial	17	134	1	14	14	180	tʃitof**u** ‘stove’
u	coronal (=alv. or post-alv)	52	25	0	0	0	77	ḅuʃ**i** ‘bush’
i,e,a,o	coronal	895	2	25	8	27	957	ejit**i** ‘eight’
i,e	dorsal	92	0	6	2	8	108	hwik**i** ‘wick’
a	dorsal	30	2	0	0	7	39	mağ**i** ‘mug’
o	dorsal	3	4	0	23	1	31	kok**o** ‘cork’
u	dorsal	1	7	0	0	1	9	ḅuuk**u** ‘book’
i	liquid	22	2	0	5	6	35	ṿir**i** ‘wheel’
e	liquid	15	0	12	19	22	68	ṿer**i** ‘veil’
a	liquid	21	8	4	0	8	41	minarar**i** ‘mineral’
o	liquid	1	0	0	44	4	49	hor**o** ‘hall’
u	liquid	1	29	1	21	4	56	fur**u** ‘fool’

Vowels inserted C__C

preceding C	following C	foll. V	# of i inserted	# u	# e	# o	# a	total	example
anything	obstruent or nasal	anything	129	0	0	0	0	129	s**i**peja 'spare'
labial	liquid	i	20	13	0	0	1	34	f**i**ridʒi 'fridge'
labial	liquid	o	0	12	0	6	0	18	p**o**rofiti ~ p**u**rofiti 'profit'
labial	liquid	e,a,u	1	86	0	0	0	87	p**u**reʃa 'pressure'
coronal	liquid	anything	43	11	1	1	1	57	ḍ**i**riŋgi 'drink'
dorsal	liquid	i,e,a	51	0	0	0	1	52	ğ**i**rini 'green'
dorsal	liquid	o	9	0	0	6	0	15	ğ**i**roṿu ~ ğ**o**roṿu 'glove'
dorsal	liquid	u	0	3	0	0	0	3	ğ**u**ruu 'glue'

4. Devise Dᴇᴘ-V constraints of varying levels of specificity to capture these patterns. E.g., Dᴇᴘ-i, Dᴇᴘ-i/[labial]__, Dᴇᴘ-i/[+round][labial]__, Dᴇᴘ-i/[+round]__, etc. You’ll have quite a lot of these. Feel free to throw in markedness constraints too if you like.

Fit a MaxEnt model

5. Construct an OTSoft-formatted tableaux file with your constraints for each of the 21 cases above. Each input should have 5 output candidates (one for each vowel). Be sure to save your file as tab-separated text (.txt), not Excel (.xls).

6. Run the tool on your tableaux file (don't forget to specify an output file too). By default, the learner has basically no smoothing term (i.e., huge σ), so it will find weights that fit the data as closely as possible. If the fit to the data is poor, consider adding more constraints.

7. Play with penalizing constraints for being complex. To do so, make a file modeled after SampleConstraintFile.txt in the MaxentGrammarTool folder that you downloaded. Each line is for one constraint; it has the constraint name, the constraint's value of μ, and the constraint's value of σ. μ is the constraint's "preferred" weight (zero by default); σ (huge by default) determines how willing the constraint is to depart from that preferred weight. A smaller value of σ means the constraint requires more evidence to depart from its preferred weight. Play around with different σ values for the constraints, to implement the idea of favoring simplicity. Think about whether certain constraints should have stricter σs than others. Make sure you're choosing σs small enough so that your results are appreciably different from what you got in the previous step. The fit to the data will be worse than in the previous step—smoothing always has that effect.

What to turn in

- Your write-up should say what constraints you used and why.
- It should include the results of your initial run with default σs: what constraint weights were learned, and what probabilities does the resulting grammar predict for the candidates? (This info is best presented in table form.) Discuss any places where the fit to the data is poor (any ideas why, or just uncapturable noise?). It would be nice to use Excel or R or the like to plot the frequencies you fed in vs. the probabilities that the grammar learns.
- Do the same for your custom-σs grammar. What σ did you give to each constraint, and why? What weights were learned? What probabilities predicted? Discuss the differences between the two grammars. Where did smoothing end up hurting the fit the most?
- I'll also provide a spot on CCLE where you can upload your four files (tableaux file, constraints file for the custom-σs grammar, one output file for each of the two grammars).
- But do your best to ensure that I won't look at those files: your write-up should have all the information I'd want.

References

Uffmann, Christian. 2007. *Vowel epenthesis in loanword adaptation*. Tubingen: Max Niemayer Verlag.