The UCLA Phonotactic Learner

Draft manual, August 2006

1. The program

- This is in Java, and consists of many dozen files with the suffix **.class** (i.e., Java executables.)
- The source code is also given, consisting of the same files with the suffix **.java**.
- The usual business about installing Java and putting access to java.exe in the Path command, is in effect.
- The Shell program (see below), automatically runs the command
    - path C:\Program Files\Java\jre1.5.0_02\bin

  But if you are running it on your own you should type this first, so that Java can be found.
- The batch file can be made with Notepad or any other simple text editor.

2. The Shell

- Bruce has written a Windows shell, which checks input files for many, many, possible errors, prepares certain files that would otherwise have to be prepared by hand, and makes prettier output files.
- Directions thus come in two flavors: Pure Java, and Windows Shell.

DIRECTIONS WITH PURE JAVA

3. The Algorithm

- includes a stochastic element: the salad selection routine starts each sequence of n salad items with a new random seed
- Hence there is no guarantee that multiple runs will yield identical results.

4. Making Sure the Program can Access Downloaded Code

- Colin uses fancy downloaded Java routines that have to be accessed when the program runs.
- This is a bit tricky because they are sitting inside .jar files, similar to .zip.
- Generally, the way to do it with a command line is to put this stuff:

```
java -Xms650m -Xmx650m -classpath uclapl.jar;colt.jar;datafile.jar;jas.jar;pal-1.5.jar;commons-math-1.0.jar;fsm.jar;trove.jar
```

up front in the command; after which comes the name of the program, UCLAPhonotacticLearner, followed by all of the Java command-line options.

5. A Sample RunMe.bat file

from Bruce's setup. This ran the learner on a Shona vowel harmony simulation. Note the path command to tell Windows where Java is.

```
cd C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Java7-21

Path C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\MyJava\bin\

java -Xms650m -Xmx650m -classpath uclapl.jar;colt.jar;datafile.jar;jas.jar;pal-
1.5.jar;commons-math-1.0.jar;fsm.jar;trove.jar UCLAPhonotacticLearner -gui -
maximumPathLength 8 -tiers
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\FeatureList.txt -phonemes
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\FeatureChart.txt -
naturalClasses
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\NaturalClasses.txt -
trigramNaturalClasses
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\TrigramLimitationFile.txt -
corpus C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\TrainingData.txt -
select -maximumFieldSize 30 -sampleType graphical -sigma2 1 -maximumGramSize
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\TierGramSize.txt -
complementClasses  -weighting ConjugateGradient -weightingIteration 10 -sampleDiameter 3 -
oeThreshold .35 -projections
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\Projections.txt -test
C:\AR\Phonotactics\UCLAPhonotacticLearner\VBInterface\Inputs\TestingData.txt -sample >
ProgramTrace.txt
```

6. Input files and parameter specification

- All files can be named whatever you please.
- The name is entered on the command line interface, preceded by a -*n* entry that says which file it is.
- You need to include a path, unless it is in the same folder that contains the program (i.e., the batch of *.class* files). The path can be relative to the main folder, if it's in a subfolder.

6.1 Obligatory Input Files

6.1.1 The Feature File

- Access it on the command line with **-tiers filename**.
- Internal format:
  - ➢ First line should be

```
Tiers                    NO COLON!
```

- Then, list the features, one per line.

- It is crucial that feature names not have spaces. Use _ or whatever.
- Java is case-sensitive.
- The first feature must always be **word_boundary**.

6.1.2 The Phoneme File

- This currently is needed, but according to the revised spec, it should merely be read off the feature chart, since it's redundant.
- Access it on the command line with **-phonemes filename**.
- Internal format: this is the format that Bruce has been passing on to Colin, viz.
  - Line 1: Segments
  - Line 2: #TAB{SPACE+word_boundary,SPACE[values for features (0/+,-)Feature,SPACE]$_0$}
  - Line 3+: SymbolTAB{SPACE-word_boundary,SPACE[values for features (0/+,-)Feature,SPACE]$_0$}
  - Colin has been replacing spaces with underscores to conform with the restriction above.
- If the file includes a feature not in the feature list, the program will merrily proceed onward, so exercise caution.

6.1.3 The Natural Classes File

- This, too, should ultimately not be an input file, but should be calculated.
- Access it on the command line with **-naturalClasses filename**.
- This is exactly the output of SimilarityByKie.exe, with two changes:
  - Remove **the final two lines** that give information about how many classes.
  - Remove the **null natural class**, which often comes up with the first privative feature in the list.
  - It appears necessary to remove the natural class [-word_boundary], i.e. no natural class that encompasses all segments.

*6.1.4 An Obligatory Parameter: **-gui***

- At present, this must be included in the command line; perhaps future versions will make it optional.
- March 2006: no longer needed

6.2 Optional Input Files

6.2.1 Trigram Limitation File (UG)

- This implements a bit of UG. It is supposed that languages cannot promiscuously use trigram constraints, but must limit themselves to a coarse subset of the feature system.
- The option is used when the user is not doing the UG herself, but is relying on the UCLA Phonotactic Learner to fabricate the constraints.
- The file is simply a list of features, each with a plus minus value, such that:

> The Learner will not use a trigram constraint, unless (at least) one of the matrices it contains is a single-feature matrix, consisting of one of the feature values given on the list.

- Hence it will normally contain "primal" features like [syllabic] and [word_boundary], or the so-called "Root" features.

- Access it on the command line with **-trigramNaturalClasses filename**.
- Format (example):

+syllabic
-syllabic
+word_boundary

- To cancel this aspect of UG (while still letting the program fabricate the constraints), place the following on the command line: **-trigramNaturalClasses all**
- If the user wishes to provide her own constraints in a file, simply omit **-trigramNaturalClasses** from the command line. (You can leave it in, but in the presence of a user-specific constraint file, it will have no effect.)

6.2.2 The Training Data

- Access it on the command line with **-corpus filename**.
- Format:
    - Just transcriptions (no annotations)
    - Phonetic symbols solely from the phoneme list.
    - Phonetic symbols are to be separated with spaces.
    - No leading or trailing spaces.
    - Frequency is designated with repetition
- If the file includes a symbol not in the phoneme list, the program will identify it.

6.2.3 Testing data

- For test using these words, access it on the command line with **-test filename**.
- The set of blick words should have the same format as the training set.
- **See detailed spec, for the need for comments on each testing items (such comments can be frequencies, or categories, etc.)**
- This will only make sense provided you have in hand (either computed, or in a file whose existence is pointed out in the command line options covered above):
    - Constraints
    - Weights

6.2.4 List of User-Defined Constraints

- from which selection takes place
- Access it on the command line with **-constraintList filename**.
- Format: see example

6.2.5 File of Evaluation Order

- Bruce never implemented this. I think it should not exist; the program should just use the order given in the constraint file.
- This determines the order in which the constraints are pulled out of their file and inspected.
- The program will keep inspecting, following this order, until it finds a constraint that satisfies the Accuracy Criterion.
- This file consists of a single column; a real number for each constraint; the program considers the constraints in *ascending* order of these numbers.

- *Do not use scientific notation.*
- Access it on the command line with **-constraintOrder filename**.
- Tiers will be considered in file order.

6.3 Parameters

6.3.1 The ***Select*** *Parameter*

- Set this parameter by including -**select** in the command line.
- It means "find the constraints, either from an input file, or from the default UG, that best model the data".
- It is left out if you are blick-testing, or if the constraint set is meant to be used completely.

6.3.2 The Parameter ***-constraints filename***

- This means, "use *all* of the constraints in the filename specified".

6.3.3 The Parameter ***-weights filename***

- Read in a list of predetermined weights, one for each constraint in the file mention in §6.3.2.
- Format: simply a column of weights.
- *These must be nonpositive.*
- Do not use scientific notation.

6.3.4 The Parameter ***-train***

- Use this in conjunction with the command line option **-constraints filename**.
- When it is present, the system will find the best weights for the complete set of constraints in the file specified in **-constraints filename**.
- Note that the presence of **-constraints filename** will *not* by itself cause weights to be learned for this file.
- Not needed if you are using the default primitive UG.

6.4 Parameter Governing the Length of the Run

- Add the command line option **-maximumFieldSize n**, where *n* is an integer.
- The current default is unbounded (keeps learning until no selected constraint can satisfy the accuracy criterion).
    - The accuracy criterion is currently set to follow a schedule, viz.

.01	where "accuracy" is defined as
.1	
.2	violations in training set/violations in current salad
…	
.9	

- If **-maximumFieldSize n** is selected, and the program reaches a point where no selected constraint can satisfy the accuracy criterion, it halts, even if *n* constraints have no yet been installed in the grammar.

6.5 Parameter Governing Salad - NOT SURE IF THIS IS OBSOLETE; ASK COLIN

6.5.1 Size

- Add the command line option **-sampleSize n**, where *n* is an integer.
- The current default is about 3000.
- In principle (no guarantees) larger salads could increase learning accuracy (and, of course, increase run time).
- The program does not give *exactly* this value, but probably something within the same order of magnitude.
  - Specifically, if the user asks for a salad bigger than the corpus, the actual salad will be of the length which the the closest integer multiple of the training set size.
  - If the user asks for a *smaller* salad than the training set, the program will return a salad of approximately that size.
  - The program creates salads that are of approximately the same length distribution as the training set.

6.5.2 Type ***PROBABLY OBSOLETE; CHECK WITH COLIN***

- Add one of the following command line options:
  - **-sampleType sequential**. This will cause the program to generate samples that match the length distribution of the training data, but are otherwise unconstrained. This seems to be the best choice…
  - **-sampleType monosyllables**. This will cause the program to use a sample-generation program that insists on there being exactly one vowel per sample item.
  - **-sampleType onsets**. This will cause the program to behave quite similarly to when you select sequential, but with different code. Stay tuned for clarification or deletion.
  - xxx graphical

- To run the **monosyllables** option, some conditions must be met:
  - There must be both consonants and vowels in the segment inventory.
  - There must a feature [syllabic] to distinguish the two.
  - The consonants (i.e. [–syllabic]) must form a contiguous bloc in the feature chart, as must the vowels. Either consonant-vowel, or vowel-consonant order is acceptable.

6.6 Assessing the result

*6.6.1 The Parameter -**sample***

- This causes the program to create an output file with salad in it.
- This is the terminal, best salad.

- Like blick testing, this needs a constraints file and a weights file.

6.6.2 Summary: How To Blick Test

you need these options to be selected:

-tiers filename	
-phonemes filename	
-naturalClasses filename	
-test filename	the blick test data
-constraints filename	
-weights filename	

7. Output Files

7.1 Grammar.txt

- This is a list of constraints and weights.
- The constraints might be concocted (using the default UG), or selected from (user-supplied UG, or just the list you gave it; see options above.
- See detailed spec for further information that should be included in this file.

7.2 BlickTestResults.txt

- For each blick word, the summed weights of the violations.
- See detailed spec for further information that should be included in this file.

7.3 SampleSalad.txt

- This is the salad created if you include **-sample**; see above.
- The salad, as well as the training data, can be resubmitted to the program as a set of blick items.
- See detailed spec for further information that should be included in this file.

8. Polysyllabic Salad Creation PROBABLY OBSOLETE; CONSULT COLIN

Word → (Syllable)*
Syllable → Onset Nucleus Coda
Onset → C_0, where C is defined in the feature system with the feature [–syllabic].
Nucleus → V_0, where V is defined in the feature system with the feature [+syllabic].
Coda → C_0, where C is defined in the feature system with the feature [–syllabic].

Grammar is statistically annotated according to training set.

Find distribution of syllable lengths of words, where these are counted by finding the number of V_1 sequences.

Find distribution of lengths of Nuclei, where these are counted by finding the number of V’s in each V_1 sequence.

Find distribution of lengths of Onsets, where these are counted by finding the number of C’s in each *word-initial* C_0 sequence.

Find distribution of lengths of Codas, where these are counted by finding the number of C’s in each *word-final* C_0 sequence.

When salads are generated, the following is done:

(a) Pick a number of syllables from the empirical distribution.

(b) Pick, for each syllable, an onset length, a nucleus length, and a coda length, from the corresponding empirical distributions.

(c) Fill these positions with consonants and vowels as appropriate, in equal frequencies.

This creates seeds.

The seeds are improved by successive migration. However, this migration is constrained: one makes one change, which consists of replacing a C in an Onset or Coda position with a different C, or a V in an Onset or Coda with a different V.

9. New Parameters 11/17/05

9.1 -maximumGramSize

This can be 1, 2, 3, or 4. The default is 3.

9.2 -weighting PROBABLY OBSOLETE; CONSULT COLIN

This has three possible values.

ConjugateGradient

This is what we had first, and as of 1/31/06, it is compulsory.

ImprovedIterativeScaling

This is out of della Pietra/dP/Lafferty. Retained for possible future use.

ObservedOverExpectedAtDiscovery

Bruce’s suggestion.
Let t be the time at which constraint C is discovered.
Then the weight of C is

$$\log(\text{numberofviolationsOfCInCorpus}/\text{NumberOfViolationsOfCInSample}_t)$$

Still works

9.3 -accuracySchedule

-accuracySchedule PathAndFileName

The accuracy schedule is an ascending list of numbers, all between 0 and 1, e.g.

```
.1
.6
.9
```

See §**Error! Reference source not found.** above for how this works.

10. Projections

10.1 Defining projections

- To define a projection, you specify feature values, *all* of which must be satisfied for a segment to be projected.
- Also, you define which features get projected.
- Thus, in the file you put, e.g.:

TierName +Feature1, –Feature2, +Feature3 : FeatureA, FeatureB, FeatureC

There *must* be a space here also here, or you will be toast.

where the numbered features are the criterion for projection, and the lettered features are the projected features.

- Examples:
  - for sibilant harmony,

    Sibilant +strident : anterior

  - For vowel harmony:

    Vowel +syllabic : high, low, back, round

    (and whatever other vowel features there are)

  - For stress, based on weight, which is based solely on vowel length:

    Rhyme +syllabic : long, stress, main

- If you are doing multiple tiers, then each tier occupies a separate line in the file, like this:

  Vowel +syllabic : high, low, back, round

Rhyme +syllabic : heavy, stress, main

10.2 Instructions for use

Command line is:

```
-projections FileName
```

If omitted, nothing is projected.

11. Method for Computing Expectations PROBABLY OBSOLETE; CONSULT COLIN

There are two ways:

A. preparing a sample of pseudo-words
B. forming a weighted finite state machine to represent the set of all possible words

Method A imposes no limit on constraint type (relevant for future development).
Method B is restricted to n-gram constraints, but is more accurate.

Right now, the code uses B for finding new constraints, and A for weighting.

So, it is important for the launching code to specify the flag

```
-weighting ConjugateGradient
```

12. Complement Natural Classes

Complement natural classes can be enabled with the **-complementClasses** command line option. This option does the following:

- For each natural classes except [+word_boundary], the learner creates a complement natural class.
- The complement ^C of class C contains all of the segments that are not in C except the word boundary <plus word boundary> to be decided
- The learner prunes every complement class that is extensionally identical to (contains the same segments as) an original natural class, given the segment inventory
- Constraints can mix original and complement natural classes freely; this makes them more powerful than traditional if ... then ... rules, because both the 'focus' and the 'context' can include complements.
- Notation: [^...]stands for the complement of class [...].