Semi-Automated Formant Extraction with Praat

A Paper on Doing Computational Linguistics

Taylor J. Meek
June 3, 2008

LING 435: Applied Linguistics
Portland State University

My choice of project centered on Computational Linguistics, and particularly the fascination I developed with Praat's role in Phonetics. Jeff Conn's (Assoc. Prof. of Applied Linguistics, PSU) dialect studies made me curious about his methods of capturing speakers' vowel inventories, and through discussions with him, came to find that the work necessary to chart these vowel inventories, such as the one in Fig. 1, represent a largely manual and lengthy process. From my background as a computer programmer in business environments, this seemed cumbersome and material ready for automation. When the opportunity arose for me to investigate and work on this problem in Applied Linguistics, I felt confident that I could achieve some measurable success.

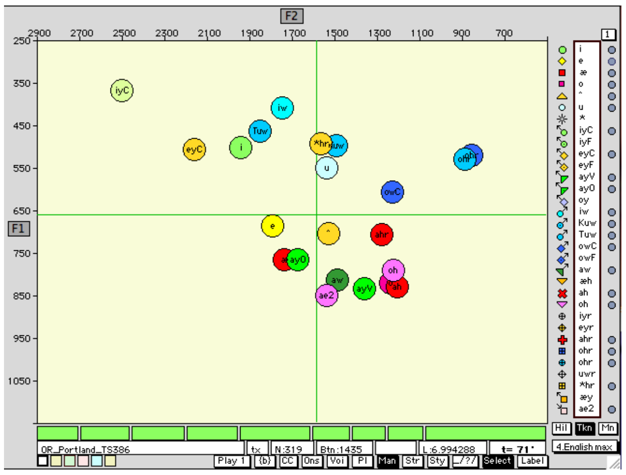


Fig. 1, Different Vowel Systems – Portland (Conn, 2008, p. 31)

As a result, I first met with Jeff Conn on April 16 to discuss his dialect research further. I followed methodologies from systems analysis to identify the key problems with the system as it stands and what an optimal solution would look like. His process for plotting vowels used Praat, a custom Praat script to export formant frequencies, Microsoft Word to edit the exported file, and Plotnik to graph the vowels. The exportation from Praat involved choosing the center of vowels based on the points that Praat draws and running a scripted command which also allowed the notation of the word and Plotnik's non-IPA-based code representing the point. This step was repeated for each file and each vowel to be plotted, followed by opening the data file in Word and editing each entry so that it matched the format that Plotnik would take in.

With the knowledge that Praat could use custom scripts, and that Plotnik utilized a specialized format for its data files, I set out to use the Praat scripting interface to export a data file that could be quickly used with Plotnik. Praat defines scripts as "an executable text that consists of menu commands and action commands" (Boersma & Weenink, 2008), and essentially allows the developer to duplicate user keystrokes and mouse-clicks automatically, enabling a user to turn what was a hundred-command process into just a few commands. The first phase to this was general research on Praat, other tools in the field, and the dialect studies that benefit from such tools in the first place. Following that I began studying Praat's scripting interface, which turned out to be a based on a fairly basic programming language. I ended up the first night of working on that with a script (Appendix 3) that takes a Sound object and generates a tab-delimited text-file based on time at regular intervals, with formant frequencies based on what Praat guesses them to be. This was something that could be opened in Microsoft Excel or other spreadsheet programs without problems for ad-hoc analysis.

The problem I immediately discovered, though, was that while Praat would display only valid formants in its internal display of a sound file, it would export formants for every portion of the sound file even when the amplitude was negligible, essentially rendering the output file with every millisecond filled with some noise or another. When graphing a sound file containing successive American English vowels enclosed by a [b], [g], [v] or [c] and a [t] or [d] (Ladefoged, Table 4.2, 2006) in Excel on a reverse axis, I was able to filter the data by cutting the data back to F1s of less than 800 Hz, and F2s lower than 2,400 Hz (based on formant frequencies representative of American English vowels (Ladefoged, A Course in Phonetics, 2006)), which cleared out a lot of the silence and also a great majority of the consonantal sounds, reducing my total number of samples from 1,967 to 881. This graph appeared to represent diphthong sounds accurately, which was very exciting, considering the Plotnik program (Fig. 1) only represented single points on a graph. These diphthongs appeared (Fig. 2) as progressive markers forming only slightly erratic lines. Since none of my data was transcribed or annotated, I have no way of identifying the source of the other points on my chart, since they could also be vagrant consonants or silence that was not removed by filtering high frequencies in F1/F2. A close examination of the chart will also reveal darker clusters of samples, which I believe may be the more fixed vowels, appearing clustered and numerous due to their relative stability over the course of a few milliseconds of speech. This chart was the end product in my 7 hours of work with Praat and Excel, and I never even got to the point of downloading Plotnik.

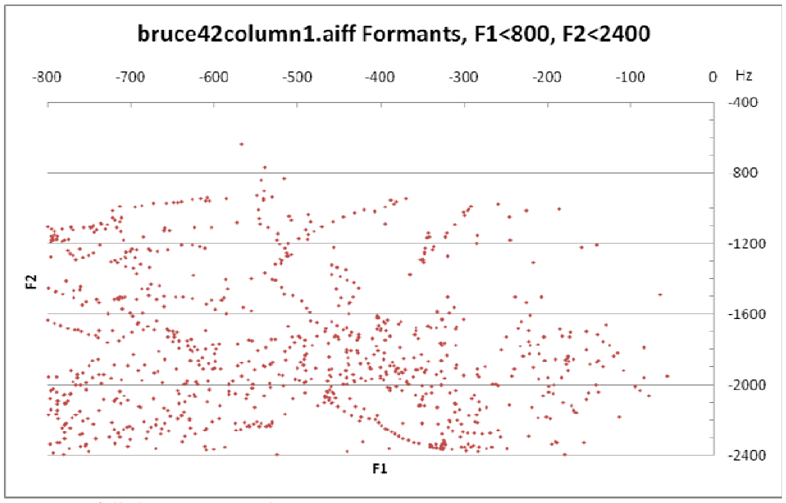


Fig. 2; see full chart as Appendix 1

There is obviously further work to be done in this process. I will highlight a number of things that I consider to be major hurdles in this process. The first, and most important of which, I believe to be Praat's "feature" of exporting formants that do not really exist, especially when compared with its own internal views. I believe that the easiest way to counteract this would be to account for the amplitude of the sound signal at the same time samples and eliminate any that fall under a certain threshold, as specified by the user. Unfortunately this will not work in noisy environments where the amplitude of the noise is greater to or equal of that of some of the speech samples, so using this tool would never be feasible in some sort of field environment. However, there is also significant acoustic research into noise reduction strategies (O'Shaughnessy, 2000, p. 415) that, while likely too complex for integration into Praat, could still yield results.

Secondly is the issue of hiding consonants from the data. In an automated method, removing samples with amplitudinally significant upper frequencies that describe fricatives or the exclusion of non-voiced sounds could be a start. Alternatively, moving to a completely different approach may be more appropriate. Since Plotnik does not represent the movement of diphthongs, and the process of extracting formants manually only takes a sample point, it may be worthwhile to combine a method of manually selecting the entire range of a vowel in Praat for exportation of its formant paths into Excel, which is capable of showing such travel.

This last suggested method would also approach a third concern of mine, and that is the lack of annotation in the Excel graph. Each point is represented identically as a red dot, which is not entirely useful for discriminating sounds from one another. By reducing the process back to a semi-manual one, these annotations could be marked during the process and once again placed on the graph. Alternatively, amplitude change recognition could be used to recognize separation between words or even some syllables prior to a complete oral closure, and then those marked sequentially as new points to consider as part of a set. This segmentation could also be done automatically with a more sophisticated segmentation system, such as those based on patterns of amplitude or spectral frequencies. (O'Shaughnessy, 2000, pp. 369, 386-402) I also conceptualized a basic method (Appendix 2) to sequential samples for their deviation from one another, and to identify contiguous sequences as those samples within a specified margin of each other, which I tested in Excel and seemed to have some limited success with, though it would be much better implemented within the Praat script itself.

What I set out to do was to create a tool that would simplify the work of dialect researchers so that they could more quickly plot speaker vowel inventories. What I ended up doing was focusing on trying to convince Praat to output clean formant samples that mean something. Plotting them was fairly easy once I got the data filtered, but even then, since I had not segmented the sounds, the data it presented was meaningless to me.

How does this work relate to applied linguistics? If Davies (2007) is correct in saying that all linguistics is applied linguistics, then whether a speech recognition tool falls into the scope of a tool for ordinary persons or theoretical linguists, it still acts as an intermediary between the fields of computer science and that of linguistics, combining the knowledge of each to produce an apparatus for making work easier or better. This too is the goal of "applied" computer science: to produce something that alleviates human work or improves upon it. Even computational linguistics has an applied computational linguistics camp, but that seems extraordinarily complicated—since after all, the only purpose of the theoretical portion of computational linguistics is to provide meaningful ideas and tools for the application side to implement problem solving solutions with. As my work shows above, that distinction becomes almost invisible because as a developer, problems will always arise, and working around those with novel approaches is the only way to get past them; if those approaches don't exist, they must be created, which requires both a theoretical knowledge of linguistics and the applied computational or other knowledge to implement your solutions.

Works Cited

Boersma, P., & Weenink, D. (2008, May 31). Praat Manual: Scripting. Amsterdam, Nederland.

Conn, J. (2008, April 22). *SPHR 370 S08 Lecture Notes.* Retrieved 4 6, 2008, from Jeff Conn's Webpage: http://web.pdx.edu/~connjc/Phnx%20S08%207%20Apr%2022.ppt

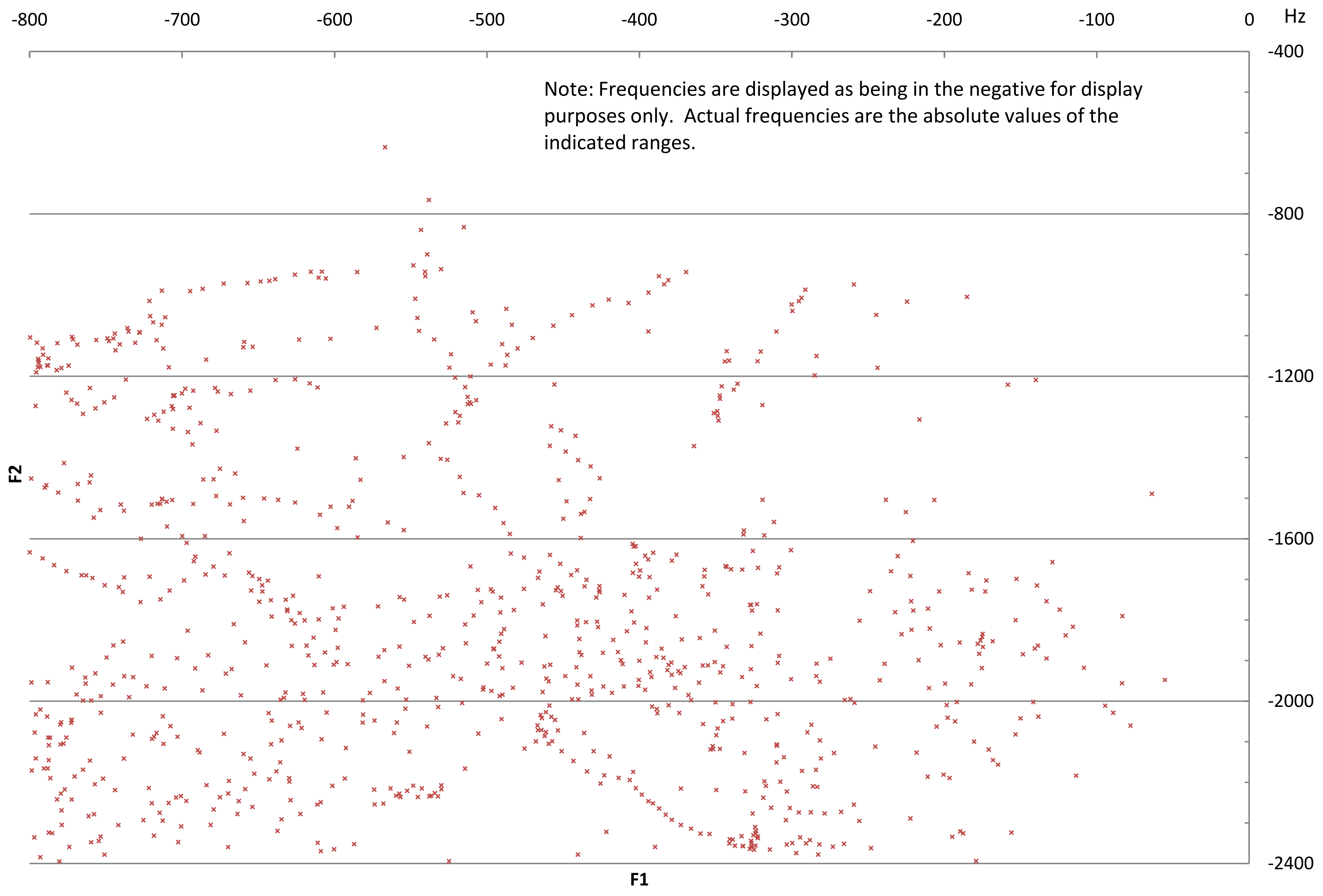
Davies, A. (2007). History and 'definitions'. In *An Introduction to Applied Linguistics* (pp. 1-12). Edinburgh: Edinburgh U. Press.

Ladefoged, P. (2006). *A Course in Phonetics.* Boston: Thomson Wadsworth.

Ladefoged, P. (2006). *Table 4.2.* Retrieved 6 4, 2008, from A Course in Phonetics: http://hctv.humnet.ucla.edu/departments/linguistics/VowelsandConsonants/course/chapter4/Bsounds/bruce42column1.aiff

O'Shaughnessy, D. (2000). *Speech Communications: Human and Machine.* Piscataway: IEEE Press.

Appendix 1: bruce42column1.aiff Formants, F1<800, F2<2400



Appendix 2

Conceptual Method for identifying contiguous progressive samples in a formant analysis.

5/26/2008

```
With a given frame,
  Look at each formant,
    And compare it to the next frame's corresponding formant,
    And if the two formants are within MARGIN-OF-ERROR% of each other,
      Mark them as contiguous,
  And if each formant is a relative contiguous match,
    Mark the entire frame as contiguous.
  Otherwise,
    Mark the next formant as the start of a new contiguous set.

With a given contiguous set,
  Identify the formant ratios for each set,
  And note the relative progression from start to finish.
```

Appendix 3

Script File for Praat Formant Exportation

Note: Using this script requires fixing the linebreaks from the published form. For a downloadable version, visit http://www.lingnik.com/projects/formantinventory/

```
! ExtractFormantsTabbedFast.praat
! (c) Taylor J. Meek, May 2008
! Uses in-built formant identification and
extraction to write the formant data to a tab-
delimited file.

! Future Improvements:
!  + Options to both Export and View.
!  + Ask for a file name.
!  + Include ratios in view mode.
!  + Limit number of formants to output.

preselected = selected ("Sound")

form Export Soundfile Formants
        comment The filename will be the name of
the source file with the extension .txt, suitable
for opening
        comment  directly with Excel or another
spreadsheet program. i.e. 'ahmed-hello.wav' >
'ahmed-hello.txt'
        comment  The file will be exported to
the directory Praat ran from.
        choice Data_Mode 1
        button Export Only
        button View Only
!       button View and Export
!       positive
Maximum_Contiguous_Deviation_(Hz) 30
        comment The following settings match
those from "Sound: To Formant (Burg method)"
dialog.
        real Time_step_(s) 0.0
        positive Max._number_of_formants 5
        positive Maximum_formant_(Hz) 5500 (=
adult female)
        real Window_length_(s) 0.025
        positive Pre-emphasis_from_(Hz) 50
endform

# To Formant (burg)... 0.0 5 5500 0.025 50
To Formant (burg)... 'Time_step'
'Max._number_of_formants' 'Maximum_formant'
'Window_length' 'Pre-emphasis_from'
filename$ = selected$ ("Formant")

if 'Data_Mode' = 1
  filedelete 'filename$'.txt

  ! Write the header row for the output file.
  fileappend 'filename$'.txt Sound'tab$'Time
  for iformant to 'Max._number_of_formants'
     fileappend 'filename$'.txt
'tab$'Formant'iformant''tab$'Ratio'iformant'
  endfor

  ! Parse each frame,
  numberOfFrames = Get number of frames
  for iframe to numberOfFrames
    time = Get time from frame number... iframe
    fileappend 'filename$'.txt
'newline$''filename$''tab$''time:6'
    lastpitch = -1
    for iFormant to 'Max._number_of_formants'
      pitch = Get value at time... iFormant time
Hertz Linear
      if lastpitch = -1
        fileappend 'filename$'.txt
'tab$''pitch:3'
        lastpitch = pitch
      elsif pitch = undefined
        fileappend 'filename$'.txt 'tab$'0'tab$'0
      else
        pitchratio = pitch/lastpitch
        fileappend 'filename$'.txt
'tab$''pitchratio:5''tab$''pitch:3'
        lastpitch = pitch
      endif
    endfor
  endfor
  Remove
  select 'preselected'

elsif 'Data_Mode' = 2

 ! Write the header row for the output file.
 clearinfo
 print Sound'tab$'Time
 for iformant to 'Max._number_of_formants'
     printtab
     print F
     print 'iformant'
 endfor
 printline
 ! Write each frame to the info window.
 numberOfFrames = Get number of frames
  for iframe to numberOfFrames
     time = Get time from frame number... iframe
     print 'filename$''tab$''time:6'
     nFormants = 5
     for iFormant to nFormants
        pitch = Get value at time... iFormant
time Hertz Linear
        if pitch = undefined
           print 'tab$'.
        else
           print 'tab$''pitch:3'
        endif
        pitch = undefined
     endfor
     printline
 endfor
endif
```