Katie Schmittlein

Ling 592

Final write-up

5/8/2018

Pitch-Tracking

When wondering what I wanted to do for this final project, I remembered during my last semester taking intro to phonetics and being introduced to praat. We were shown the different ways in which to use the program, one being finding the pitch. However, it became clear that whatever method praat uses to calculate pitch, or more accurately- to calculate f0, it was not perfect and made mistakes that were clear when analyzing the sound compared to its pitch track. From this point on I wondered how exactly not only praat, but any pitch tracking program works, and what are the difficulties in making such a program. It seemed to me that finding a correct f0 track would be difficult, if just because of the variation in human speech quality and the fact that it must change very rapidly and often. For these reasons I wanted for this project to be my way of researching and discovering just how pitch tracking works. When suggested to me, working with a code already written and working to understand it seemed like the best approach to go about doing this.

David Talkin’s paper on RAPT, a robust algorithm for pitch tracking, was the best insight I got into the difficulties and various methods used by different pitch trackers, as well as his reasoning for choosing the methods he had. Before looking at the code, I made sure to read through this paper the really understand what is being accomplished in the program step by step. It also gave helpful background information that aided in my understanding of pitch tracking methods as a whole.

Some challenges that this paper brings up when creating a pitch tracker are that F0 changes rapidly, even as often as each glottal period, strong sub-harmonics can create a large discrepancy between the best estimated f0 and what is actually perceived as pitch, difficulties in discerning voice onset and offset times makes it difficult to calculate accurately, and it can be difficult to discern background noise from breathy speech. All of these aspects are working against whoever attempts to make a program to track the f0 of a sound file. It stands to reason with these difficulties present that the tracker created would not be 100% accurate all of the time, nor would it always correlate exactly to what someone hearing the sound file would identify as the pitch. These aspects gave me a lot of insight into the challenges behind pitch tracking and knowing these I believe will help in the future while using praat. With an understanding of the difficulties of any pitch tracking program, I believe it will be easier for me to understand some of the strange f0 contours that can be seen with certain types of speech qualities.

The paper next goes into the steps that most pitch trackers use in calculating best estimates for f0 and the variations that are present. This section was especially helpful in me understanding the code and what was trying to be accomplished in each step of it. The first step in most pitch tracking programs is pre-processing, where the speech data is altered to be better analyzed. With RAPT, the goal was to have both accuracy and ease of processing for the program. Instead of using other methods such as low-pass filtering and flattening the signal spectrum which are used to try and make the ‘true’ f0 more apparent, this program in the pre-processing stage stores two different versions of the speech data, one at its original sample rate, and one at a reduced sample rate. The idea here is that the program can then, in the second stage of the program, pass through the sound file with a reduced sample rate first, picking out the local peaks that stand out without having to process all of the data from the speech data at its full sample rate. The second pass is then through the sound data at its original sample rate, only looking at regions around peaks already found in order to get a more precise estimate.

There are many methods that this paper describes for doing this f0 candidate selection process, including one method we learned in class, using the cepstrum. Even though this particular program does not use this method, I am glad it was included in this paper because with the knowledge I already had from this class, this was the best way for me to understand at least the basics of what this stage of the program is trying to accomplish. The method that RAPT does use is the normalized cross-correlation function, or NCCF; it is shown in this paper that this method provides more accuracy in finding the true period of a ‘problematic’ speech signal by looking at the location and height of maxima at each frame.

The final step in estimating f0 is deciding which of the candidates that the second stage picked out is the ‘best’ estimate. In RAPT the method used is dynamic programming (DP), which can increase the accuracy of f0 estimation by effectively increasing the sample rate around the peak and relocating the peak at this higher sample rate, and chooses the best set of NCCF peaks that fit several aspects of the speech quality of the data.

After obtaining this information about both RAPT and pitch tracking programs in general, I moved on to actually analyzing the code in matlab in order to rewrite it in python. Because I like writing notes on paper and highlighting things by hand my first step was to actually print out the code to work with. Using this I went in and wrote notes on what is happening at each section of the code, as well as marking functions or other aspects of the syntax that I did not understand. The information given to me from the paper was very helpful here because it gave me a better understanding of what was going on in the code; even though the comments in the program define what is happening each step of the way, without the background knowledge from the paper I don’t think I would have been able to figure out what the program is trying to accomplish in a broader sense.

My next step was to actually work with the code in matlab to understand the syntax and functions that I was not familiar with. It was here that my limited knowledge of both C and python became the most of a hindrance. The functions I was unfamiliar with were easy enough to understand using matlab a bit, and were easy to find an equivalent in python. Here again I took notes on the printed out code so that I would remember which function in python matches the function in matlab. The syntax was a lot harder for me to figure out. There is not as clear a way to find information regarding the correlation between syntax used in matlab and what is used in python and this proved to be the biggest challenge when it came to writing the code in python.

When actually doing the conversion from matlab to python, I decided to not copy and paste the code into a jupyter notebook and fiddle with it, but to instead rewrite it completely, using the original code as a guide. In order to be able to easily refer back and forth between these two versions of the code, I kept the comments exactly the same so as to give myself an easy way to find the same spot in both programs.

Although the syntax conversion was difficult at times for me to figure out, I feel this project was a success for me in furthering my understanding of writing programs, fixing bugs, and understanding code in a new language, as well as understanding the process that lies behind pitch trackers such as the one used in praat. Through this project I have certainly improved in my ability to go into a program that is new to me and piece by piece work to understand what is being accomplished at each step. It also gave me a bit of knowledge about how the C language works and some ways in which it differs from python. The thing I am most happy about taking away from this project is the understanding I now have about the workings of pitch trackers. I love understanding how things work and the pitch tracking in Praat has had me curious about this particular aspect of speech processing for a while now. This may not be the same method exactly that Praat uses in order to provide an estimate for f0, but understanding this code and the different methods described in the paper gave me some much valued insight into how pitch tracking works, the difficulties of providing an accurate estimate, and how discrepancies between an estimated f0 and the perceived pitch can come about. The converted code I have written up may not be perfect but I still feel this project has been a success in my own personal growth in programming and growth in my knowledge of pitch tracking. Although I am certainly still a novice at both python and C, this has helped in me seeing that understanding code, even code in an unfamiliar language, is not as daunting as it may seem.