Computing BEng

Individual Project

Rubato: An Adaptive Musicality Tutor using Pitch Analysis

Conrad Godfrey

Supervisor:
Iain Phillips

June 15, 2014

Contents

Chapter 1

Introduction

In this paper I propose an application to harness the power of the modern web browser to deliver engaging musicality tutoring based on adaptive learning principles. The application will comprise of various exercises designed to challenge and improve a user's musicality, and keep track of their progress as well as adapt to their strengths and weaknesses in different areas.

As usage of the world wide web has become ubiquitous among citizens of the developed world, web based teaching has grown rapidly to try and modernize learning methods to fit into the 21st century lifestyle. Increasing numbers of companies like Coursera,[1] Codeacademy,[2] Duolingo[3] are providing easily accessible education for anyone with an internet connection and a desire to learn. Codeacademy and Duolingo in particular are notable for their use of rich, highly interactive learning tools which - through requiring the user to have an input into the educational process - establish a feedback system with powerful results.[4] These companies offer a variety of different programmes the user can study, and each of these is taught through a series of exercises, the results of which are used to track the users progress through the course, and provide statistics on how much they have learned. Duolingo, a website that teaches foreign languages, takes it a step further by introducing adaptive learning methods to recognise the user's proficiency in various areas, information it can then use to tailor-make lessons to fit the user's needs. Duolingo is arguably the best best-known example of an adaptive learning system put into practice. It will therefore often be used for comparison throughout this report, as it has also provided a lot of inspiration for my thinking about the direction my project will take.

The web is a great place to teach music theory, and train the musical ear, a pair of skills we shall refer to under the umbrella term of "musicality" from hereon in. The rich media options possible on modern day web browsers mean that the input and output of music to a web browser is not only easy to implement, but easy to make user friendly. However, while there are many "music theory tutors" and "ear trainers" available, there are no existing adaptive learning solutions. Such a solution would hopefully be invaluable to potential learners as the current best solutions still have no idea who you are and what you've achieved after you leave the page.

In order to teach adaptively we must analyse information about a user's performance, and then change how we teach accordingly. There are various ways of measuring musicality, which will be discussed in more detail later, and we will rely on these metrics to adapt the learning experience to fit the user.

1.1 Objectives

I aim to build a program that can successfully teach musicality. To ensure I achieve this goal, the following criteria must be met:

- **Adaptive** - The product must analyse the user's progress and their strengths, and adopt the content of their learning experience accordingly. This will require a user account system, as well as intelligent handling of the exercise data they generate.
- **Intuitive** - The product must be simple to use, and the exercises given to the user must be simple to understand, even for a beginner.
- **Engaging** - The user must want to learn and continue learning. In a study of Duolingo's effectiveness, 93.8% of participants intended to continue using the website after the study had finished.[5] I would like to aim for at least 75%.
- **Pitch Recognition** - Another way I'm aiming push the boundaries of musicality teaching is through introducing exercises that involve pitch data to be submitted by the user, and then grade them based on how accurate they are.

Chapter 2

Background

2.1 Adaptive learning

Web based learning has been a topic of interest almost since the birth of the web,[6] and the introduction of the HTML5 standard has opened up even more opportunities for computers to play a role in the education process.[5] 3 notable examples of web based learning systems (WLS) are:

- Coursera, a site that provides lecture material for various university level courses.
- Codeacademy, a site designed to help people learn coding by getting them to undertake interactive browser based programming lessons.
- and Duolingo, a site designed to help the user learn foreign languages through interactive exercises that tries to tailor-make each exercise to the users needs by analysing their strengths/weaknesses.

The end aim of these sites is the same as all WLS's: to educate the user, but there are key differences between them which we can use to categorise them further into a hierarchy.

- Coursera is a *static* WLS, it allows a one way interaction whereby the user can view/download learning material. While this is surely useful, it is really the base level of what a WLS can do.
- Codeacademy is a *dynamic* WLS, it allows a two way interaction that introduces feedback for the user, enhancing the learning experience beyond a static WLS.
- Duolingo is an *adaptive* WLS, it includes all the features of a dynamic WLS, only it can treat every user differently by analysing their strengths and weaknesses.[7] This adaptive approach allows for superior teaching to static or dynamic WLS's, as if implemented correctly it will start to mimic the tailor-made learning experience one might receive from a real-world 'for-hire' tutor.

2.1.1 Adaptive learning in music

There are many ways that adaptive learning techniques can be applied in a musical context. As an example, imagine a simple exercise.

- The user is played a rhythmical phrase.
- They must then replicate the phrase by tapping it in on the space bar.
- The application then calculates how accurate the user's approximation of the rhythm is, and feeds this information back to the user

After repeating this exercise multiple times the application notices something: The user is always getting examples featuring multiple consecutive dotted quavers wrong [1], and determines that this particular rhythmical device is something the user is struggling with. It can then subtly adapt future exercises to incorporate this device prominently, in order to expose the user to it as much as possible, and hopefully cause them to improve their understanding of that specific rhythm, and rhythm in general.

2.2 Platform choice

As the modern internet browser has become more and more powerful, web apps have been able to reduce the previous speed disadvantages the faced when compared with their native equivalents. The web-platform is advantageous due to having a singular codebase, meaning that it can be used by anyone with a web browser, independent of their device, meaning it has the potential to reach more users. Another key advantage of building a web app is that as user testing is so key to evaluating the app's success, when the time comes, and I want to show it to users, I can easily point people to my website and people will know how to get to it. Distributing a mobile app to others for testing purposes is painful, and difficult to update once they have installed it, with a web app I can instantly change the build of my website, and my aunt who's testing it in Jamaica won't have to do anything more than refresh the page.

2.3 Existing Musicality Tutors

There are a wide variety of existing web-based applications that teach musicality, to varying degrees. There are programs to allow you to practice interval recognition,[8] identify a note in a chord, practice recognition of rhythms.[9] A lot of these programs are standalone, and focussed on one specific area. Musictheory.net[10] is a good example of a website that that goes beyond that, it provides exercises that test a wide variety of skills, trains your ear as well as providing music theory lessons. However Musictheory.net

[1]This is a type of rhythm that could prove difficult for the user as it can sound like the rhythm is going in and out of time due to it's naturally syncopated nature against a four four baseline

provides no adaption to the user, the furthest it goes is allowing you to specify what you want to be taught, but it is not intelligent enough to work it out itself.

2.4 Music Theory

In order to understand some of the workings of the program it will be necessary to delve briefly into a little music theory. We will examine the fundamentals of Western music theory, as well as theory of pitch. From here on we shall talk exclusively in terms of Western music theory, as it is the theory that the overwhelming majority of contemporary Western music is based upon.

2.4.1 Pitch

The fundamental quality of a note played on the piano, plucked on the guitar, or sung by the voice is it's pitch. The other defining quality of a note is it's timbre/tone i.e. whether it sounds harsh or mellow, buzzy or clean, but this is a quality that is very hard to quantify, and also not as important to the overall melody of a piece of music - The same melody played on a guitar and a piano can certainly be considered to be the same piece of music, despite the differing tones of the instruments, but if you change the pitch of any notes, it becomes a different melody entirely. So of an individual note in a melody, we can say that it's pitch is it's defining characteristic.

What do we mean by pitch? Scientifically, the pitch value of an audio signal is determined by that signal's fundamental frequency, where a higher frequency is a higher pitch, and a lower frequency a low pitch, however in terms of human perception, it is not as concrete as this, as there are various different psychoacoustic phenomena that determine how "high" or "low" a given note sounds. For example, a sinusoidal tone played at the same frequency at a low volume followed by a high volume will appear to be playing 2 different pitches, with the louder tone sounding lower in pitch. However, sinusoidal tones are much simpler than the rich complex tones of a musical instrument, which due to being made of many different frequencies can provide the listener with more cues as to the pitch they are at, so for the purposes of this project we will define pitch as it is defined in most musical contexts, as a logarithmic function of frequency.

The A above middle C on a piano (A4 in scientific notation) is defined as being 440Hz, and this gives the basis for all other pitches to be defined.

Some instruments like pianos and guitars have predetermined pitches that the instrumentalist can produce. Others, like the violin or human voice, can create any pitch within a given spectrum in a continuous fashion. This presents a challenge to musician's both new and old

...to be completed

2.4.2 Human vocal range

Male and female voices can generally be split into 4 types which are determined by their singing range. They are:

- Bass: E2-E4
- Tenor: C3-C5
- Alto: F3-F5
- Soprano: C4-C6

2.5 Pitch Detection

Pitch detection is a well understood problem that has been researched for many years. The pitch of a note, as described above, is the human perception of how high or low it is. The pitch value of a sound signal is determined by the fundamental frequency, f_0 of the signal, and thus the problem of pitch detection of a signal is analogous to finding that signal's fundamental frequency. The mapping of frequency to pitch is determined by the 'tuning system' you use, and in Western music, equal temperament is the most commonly used. The way it works is as follows:

- An f_0 of 440hz corresponds to an A above middle C, otherwise known as A440
- Doubling the frequency causes an increase in pitch of an octave, so A880 is the next A above A440.
- Since a leap of 12 semitones is caused by a doubling of the f_0 , it follows that the frequency ratio between each semitone is $2^{\frac{1}{12}}$ or about 1.059.

So after we've calculated f_0 it is simple to calculate the pitch. If we assign an arbitrary numerical value to A440 such as 69 (as used in the MIDI specification[2]), then we can use the following equation to determine the pitch of a given frequency, f, as a value in pitch space, p, where semitones correspond to a gap of size 1.

$$p = 69 + 12 \times \log_2 \frac{f}{440Hz}$$

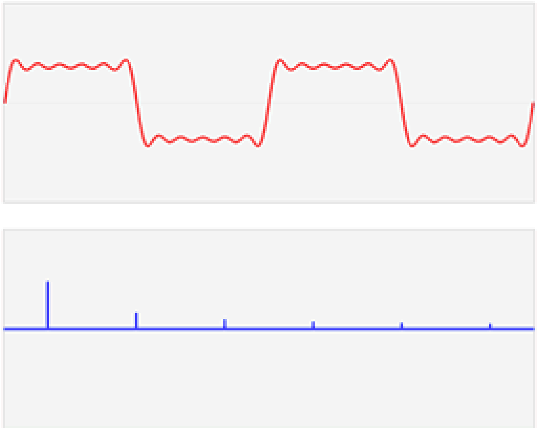
2.5.1 Fundamental Frequency Detection Methods

f_0 detection is a difficult process, and there is no 'best method' so to speak, each approach has its drawbacks and advantages, the normal tradeoff being that a method that is fast may not be reliable and vice versa. There are two approaches to f_0 detection, analysing the signal in the the time-domain, or the frequency domain. To analyse in the frequency

[2]MIDI is an industry standard for storing musical sequence data

domain requires the use of the Fourier Transform, and is therefore a slower process, but provides more accuracy. However, as we are only doing monophonic pitch detection, and responsiveness is key, then for our purposes, time-domain algorithms will be the most appropriate.

Figure 2.1: Time domain (above) vs frequency domain representations of the same signal



Autocorrelation

A standard method of time-domain f_0 detection is autocorrelation. It exploits the fact that a periodic or quasiperiodic waveform such as a sustained sung note will be self-similar by the definition of periodicity. If we compare a waveform with a copy of that waveform offset by the period of the waveform i.e. ${f_0}^{-1}$ then we should expect to see a strong correlation between them. So autocorrelation works by iterating over all the possible offset values, and determining which one has the best correlation. This correlation value for each different offset is described by the equation below, where x is the signal function, N is the window size of the waveform you are considering, and v is the offset value.

$$R_x(v) = \sum_{n=0}^{N-1-v} x[n]x[n+v]$$

Limitations

One fundamental limitation of Autocorrelation is that octave errors are frequently encountered. This occurs when the offset is calculated to be double what it should be, and is prone to happening as the nature of a periodic wave is such that if you shift it along twice the offset then you will also end up with a valid offset.

2.6 Audio processing

Chapter 3

Implementation

3.1 Web Application

The program is taking the format of a web application with different musicality exercises on them. Users can pick and choose whatever exercises they would like to complete, which are grouped by category.

3.1.1 Application flow

Django web framework

Views

Models Django interfaces with the database through Models. An example of this is the IntervalScore model below. This particular model stores all the information about a user's attempt at singing an interval in the interval training exercise.

```
class IntervalScore(models.Model):
    interval = models.ForeignKey(Interval)
     timestamp = models.DateTimeField(auto_now_add=True)
     score = models.FloatField();
    user = models.PositiveIntegerField();
    def __str__(self):
         return " Score: " + str(self.score) + " for" + self.interval.name + " with user
```

The interval field represents the interval that the user has attempted, the timestamp tells us when it was attempted, and the score field tells us the calculated score for that attempt (it will be explained how this metric is derived in ”Measuring user ability”)

3.2 Pitch Detection

Pitch detection is an important part of the exercises, and as such, robust pitch detection has been an important feature in the development of this program.

Most existing pitch detection algorithms are based on real-time applications, i.e. where audio captured from a microphone is analysed and the pitch is displayed as a constantly updating function of the audio signal. This is useful for applications like guitar tuning, where you need near instant feedback, and you can be sure that the pitch you are inputting is fairly constant.

With analysis of pitch in the human voice however, the problem is that even with well trained singers, the pitch of a note can vary quite considerable over a 2 second period, due to vibrato[1], or other causes that can be hard to pick up just by listening to yourself, so it is not always trivial to determine one pitch to summarise a sample of a single held note, and indeed, often not appropriate to do so if the singer cannot even stay on the same note for that short period of time, and wobbles around the note inconsistently.

It is with this in mind that I have developed a Segmented Pitch Detection Algorithm, that can accurately determine the pitch of a 2-3 second sample of singing.

3.2.1 Segmented Pitch Detection Algorithm

The principle of the Segmented Pitch Detection Algorithm is relatively simple: A given audio sample is split into several segments, pitch detection is run on each segment, and then

Choosing a segment size

Most PC microphones sample audio at either 44,100Hz or 48,000Hz, as these ensure that the full 20KHz spectrum of human hearing can be recorded due to the NyquistShannon sampling theorem (see Background section 2.5). For the sake of simplicity we will assume a sample rate of 48,000Hz for this report, though the program can handle either rate.

The human vocal range defined in section 2.4.2 was E2 - C6, which corresponds to a frequency range of 80-1000hz.[11] To calculate the number of samples at either end of the range:

$$48000 \div 96 = 600 \text{ samples}$$

$$48000 \div 1000 = 48 \text{ samples}$$

In order for our autocorrelation algorithm to work, it is necessary to take a segment size at least twice that of the lowest frequency we wish to sample. This is because in autocorrelation, we must compare the signal with an offset version of itself, and this

[1]Vibrato is the musical technique of intentionally oscillating the pitch of the note around a fixed point.

score is maximal with an offset equal to the period of the signal. Therefore, in our example, if we use anything less than a 1200 sample size and are trying to detect a 96Hz signal, we won't even be able to autocorrelate on a full period. [DIAGRAM] I have therefore decided to use 1200 as the segment size. Ideally this would be larger, but then you are granularity of change of pitch, as this means less windows analysed per second.

3.3 Measuring user ability

Discussion of the measuring of user ability for the various different exercises.

3.4 Adapting to user ability

Discussion of the adaptive element of the program for different exercises.

3.5 Exercises

Chapter 4

Development Tools

In this chapter I describe the libraries used to build my application

4.1 AudioContext Web API

4.2 MIDIjs

4.3 RecorderJS

Chapter 5

Evaluation

In order to evaluate the success of my program, I have used both qualititative and quantitative methods which I will outline in the chapter.

5.1 Adaptivity

How well I have achieved adaptivity in my teaching will be a difficult thing to test quantitatively, as there is no existing way of measuring how adaptive a learning system is. One way to gain a greater understanding of how effective the adaptivity is would be to use the program and artificially fail parts of exercises to try and prompt the system to adapt to my behaviour, and see how it responds.

5.2 Intuitiveness

The intuitiveness of my program is something that can only be judged by other people using it. I intend to involve others in the design process from the start, by using techniques like hallway testing to quickly get feedback on how users approach my app, and what they would change about it. At the end of the project I will also hopefully get as many different people as I can to use it, and fill in a survey at the end of their usage period detailing their experience with the app.

5.3 User Engagement

This will be handled by users answering questions about whether they would continue using it, and how much they enjoyed the process. These questions will be found on the survey described above.

5.4 Teaching Proficiency

To measure the general success of my product as a teaching tool, I shouldn't actually have to do too much, as (if succesfully implemented), my app should be able to track the users progress, and store information about how much they've improved.

5.5 Pitch Detection

I have developed a test set for my pitch detection algorithm. This consists of multiple different samples of 1 or 2 seconds of a note being sung at a specific known pitch. I have included samples from many different people in my test set to make sure my pitch detection works for as many different voice types as possible. I have also recorded people using a very basic microphone used on my laptop, so I can ensure that the pitch detection will work even with fairly low quality audio.

Here are the results of what I found...

following sections includes pictures of my pitch detection output, as well as percentage correct results for my algorithm.

Chapter 6

Plan

So far I have begun the design of the website, as well as started to implement the first pitch detection function. I am aware that I have a lot left to do, so it is crucial that I plan my time correctly

- 3rd of March - I will aim to have the basic skeleton of my website working including user log-in, and a rough pitch detection algorithm in place.
- 14th of March - Improve accuracy of pitch detection algorithm, and implement it into a ”Note from Chord” exercise.
- Suspend progress until end of term on 28th due to exams!
- With the basic web interface designed, create multiple different exercises over Easter break until 26th of April and start getting users to test them. Start to experiment with creating adaptability in exercises to see how it can be done.
- 28th of April - REALITY CHECK, see where I've come, and where I still need to go. Re-evaluate rest of plan up until June accordingly.
- 1st of June Begin intensive user testing to gather information for my evaluation write up project simultaneously.
- 17th of June Final Report due.

6.1 Extensions

If I finish ahead of time, there are a couple of extensions that come to mind.

- Support for MIDI instruments. There are a number of existing instruments you can plug straight into a PC using the MIDI to USB interface, it would be interesting to see how I could incorporate that into my project.

- Segmented crowd-sourced music transcription. Allow users to to test their music ability by playing them a short clip of music and getting them to supply information they've perceived about the segment, chord data, time signature data, melody data etc. The idea is that if enough users completed segments of a particular song, the song's structure could be pieced together.

Bibliography

[1] "Coursera, take the world's best courses, online, for free.." http://www.coursera.org/.

[2] "Codeacademy, learn to code interactively, for free.." http://www.codeacademy.com/.

[3] "Duolingo, free language education for the world!." http://www.duolingo.com/.

[4] R. Vesselinov and J. Grego, "Duolingo effectiveness study," *City University of New York, USA*, 2012.

[5] L. S. Griffiths, R. Ogden, and R. Aspin, "A profile of the future: what could html 5 do for he by 2015?," *Research in Learning Technology*, vol. 20, 2012.

[6] S. Alexander, "Teaching and learning on the world wide web," in *Proceedings of AusWeb*, vol. 95, p. 93, 1995.

[7] "Duolingos data-driven approach to education." http://blog.duolingo.com/post/41960192602/duolingos-data-driven-approach-to-education. Accessed: 2014-02-20.

[8] "Interval ear trainer - recognise the interval between two notes.." http://www.8notes.com/school/theory/interval_ear_trainer.asp,.

[9] "The rhythm trainer - rhythm recognition and representation.." http://www.therhythmtrainer.com/,.

[10] "musictheory.net, on online music theory tutor and ear trainer." http://www.musictheory.net/.

[11] "Table of note frequencies.."