Technical Report of the First Phase of the Project

Pierre-Yves Hervo
Polytech Nantes High school of Engineering

11th of July 2013

Abstract

This report explain how I managed to interface Praat and Java and how the Genetic Algorithm is implemented. The first part of this report will be an overAll to explain general concepts and the second part will deals with my particular Java implementation.

Important: This solution is working on windows 7, it is supposed to work on Linux but it won't work on a Mac because of the Praat's API.

Contents

Part I

Overall

Chapter 1

About the Genetic Algorithm

This chapter deals with the basics of how my genetic algorithm work. I will explain my implementation in the second part.

1.1 Principle

The Genetic Algorithm I use is based on the common genetic algorithm and I introduce some specificities in the fitness function. This section will present the common point with a classical GA and I will present the specificities in the next one.

1.1.1 The target

The best way to recognise a vowel is to compare the formants of the candidate sound to the formants of a well known sounds.

For example, if we want to know if a sound is a "i", we will compare the formants of the candidate sound to the values which are defined for a i.

We declared two formants for each sound. We choose two because some vowels only get two formants and if we look for three, Praat will produce an error. Each of this formant get a value for the frequency and a value for the bandwidth. Praat can easily calculate such data but we still haven't find a way to calculate the amplitude.

But this value are just a definition, in practice, we can estimate that a formant value can increase or decrease over 10% of this value. That's why

the fitness function of my GA compare the values of the formants of the candidate to the +/-10% values of the target's formants.

1.1.2 The candidate

We will compare the candidates formants to the reference formants. For that purpose, we need to synthesise the sound and calculate its formants. The software Praat will do both for us. The only thing we need to do is to send it a script with values for each parameter.

The problem of how to send this script will be solved in the next chapter.

There is 29 parameters for voice synthesis in Praat. We use a linear voice synthesis which means that each parameters get two values to represent time evolution: one for the beginning 0.0 and one for the end,here 2.0.

```
# ------------------------------------------
# Control glottis
# ------------------------------------------
# Glottal closure
Set target . . .  0.0  0.5  Interarytenoid
Set target . . .  0.5  0.5  Interarytenoid
#
# Adduct vocal folds
Set target . . .  0.0  1.0  Cricothyroid
Set target . . .  0.5  1.0  Cricothyroid
```

Figure 1.1: Example of Praat script to set two variables. Source : [Mir02]

So there is a lot of value to define. Fortunately, Praat will automaticly complete the values for some parameters by default, we will only need to define the 10 most important.

Which means that the structure we will make evolve int the GA is a structure containing 20 values, one for each parameter to set in the Praat script.

1.2 Particularities

The particularity of my Ga is located in the fitness function. I used another program (Praat) to execute a script. This script will generate a sound, analyse it to get the formants and then get the values back. Then I will compare the formant in the result from praat with the formants given to the program as the target.

It is different from other fitness function that typically take care themselves of the calculation part. Here it delegate it to another program and only do a comparison. It do it for each individual of the generation's population.

1.2.1 Evolution Operators

We use two common evolution operators for a GA : a one point cross over at each generation and then a mutation with a probability of 0.2 to append.

Chapter 2

How to connect Java and Praat

Giving the Praat's API, we need to use two different ways to connect Java and Praat. We need to consider one way to communicate between Java and Praat to send and execute the Praat script and another to send the response from Praat to Java.

2.1 From Java to Praat

We want to send a Praat's script to Praat and have it executed. The way I choose is the software SendPraat[Dye13]. It is a program developed by the same authors as Praat. It allow to send orders to a **running instance of Praat**[1]. It means we need two programs :

1. a normal Praat software already launched.
2. SendPraat which will give it orders. No need to launch this one, it only works in command line.

If you give SendPraat the name of a script, it will made Praat launch and execute it. The only thing left is to make Java executed SendPraat. For this, I used the Java Runtime Environment which can use the command line of windows. I will explain it in the second part of this report.

and praat link in bibliography+ url

Note : I used a SendPraat.exe as I used windows but you can compile the source code yourself to use it in your own operating system. If you want

[1]This is very important, if there is no Praat already launched, SendPraat won't work.

to make Praat communicate with a C program, you can use the SendPraat directive, no need to compile source code. For more information, look at the Praat's API, section Praat scripting. As I was working in Java, the solution I presented is currently the best.

2.2 From Praat to Java

There is only one way to make Praat communicate with another program, whenever the language is, the sockets[2].

Sockets are a way use in computer science to make two different program communicated. For this, they will use the network principles and send network packets to a computer on a specified port. It is not necessary that it was another computer, it can be the same and int that case, we use a local network call localhost. The first program will send a message to the other specifying the port and the second one will listen will listen to the port and get the message when it arrived.

Praat allows to send sockets by the directive "sendsocket" but it cant received sockets from another program. That is why we got to use the SendPraat program in the other side. If Praat send a socket then our GA will need a functionality which always listen to this port and will take the message. Such functionality basically call a **Server**. In that purpose, I implement a Java server that listen to a specific port. I will describe it in the second part of the report.

2.3 Recap

So in conclusion we had to consider two different side for the communications : the message from Java to Praat and the message from praat to Java. On the first side, we had to use a specific praat program called SendPraat and on the other side we had to use a Java's Server to listen to Praat's sockets.

The figure below show how it works :

[2]It only work for windows and Linux, it is the Praat API wich manage it like this.

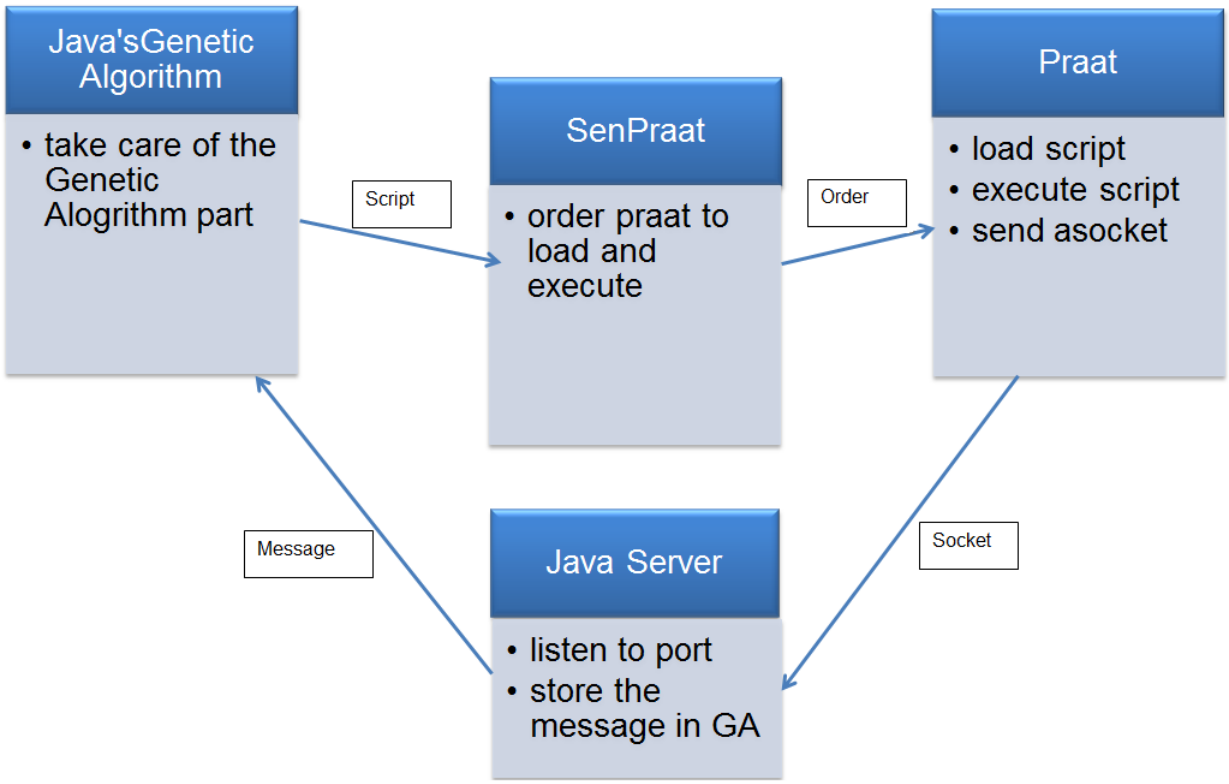

Figure 2.1: The exchange of information between Java and Praat and the role of each entity

Part II

Implementation Details

Chapter 3

the GA

3.1 watchmaker

3.2 basics elements to manipulate

3.3 the result

Chapter 4

the communication

4.1 call to sendPraat

4.2 the server

Appendix A

A scheme

List of Figures

List of Tables

Bibliography

[Dye13] Daniel W. Dyer. Evolutionary computation in java. a practical guide to the watchmaker framework, feb 2013. http://watchmaker.uncommons.org/manual/.

[Mir02] Eduardo Reck Miranda. *Computer Sound Design: Synthesis techniques and programming (Music Technology)*. Focal press, sept 2002.