Introduction to Artificial Intelligence Assignment 2

Anton Nekhaev BS21-07,
a.nekhaev@innopolis.university

Fall 2022

1 Manual for running my program

To begin with, you need to install requirement libraries (or take them from my github repository, link bellow) for execution of my program, here are they:

mido==1.2.10
tqdm==4.64.1

Tqdm just a library for fancy status bar.

Also, I think that should be mentioned that the program was tested on Python 3.11, but it should work on at least Python 3.9, given that all requirements are installed.

I decided to make my program as a console tool to make accompaniment for the melody.
So, to run it, you can simply type in you console:

```
python3 AntonNekhaev.py {name_of_source_file}.midi
```

This command will produce output with standart name.

Also you can specify several option parameters. List of them is below:

Listing 1: Output of python3 AntonNekhaev.py – – help (optinal part)

```
-h, --help            show this help message and exit

--population POPULATION, -n POPULATION
    Provide the size of initial and successive populations (default: 600)

--iterations ITERATIONS, -i ITERATIONS
                      Provide the amount of iterations (default: 100)

--out OUT, -o OUT     Name of output file
```

2 Key detection algorithm

To define a song key I use a Krumhansl-Schmuckler key-finding algorithm according to [1]. The idea behind it is that for a given key some notes are played more often that others. So, for every song I am calculating duration of every note in whole melody. And then find the correlation coefficient, which indicates the linear relationship, based on this formula:

$$R = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \cdot \sum_{i=1}^{n}(y_i - \overline{y})^2}},$$

where x_i - duration of i note, $\overline{x}$ - mean value of durations, about y and $\overline{y}$ will be described below.
In the key-finding algorithm y represents a profile of a major key or a minor key. I use different values than [1] because values in the article produce the wrong result for `barbiegirl_mono.midi`, used by me values we can be found below:

Table 1: Major profile.

do	do#	re	re#	mi	fa	fa#	so	so#	la	la#	ti
17.7661	0.145624	14.9265	0.160186	19.8049	11.3587	0.291248	22.062	0.145624	8.15494	0.232998	4.95122

Table 2: Minor profile.

la	la#	ti	do	do#	re	re#	mi	fa	fa#	so	so#
18.2648	0.737619	14.0499	16.8599	0.702494	14.4362	0.702494	18.6161	4.56621	1.93186	7.37619	1.75623

According to [1], by combining the pitch class values with the key's profile data, the key-finding algorithm determines a correlation coefficient for each potential major and minor key. Then it is simple to find the key of the song, it will have the greatest linear relationship among all others keys.
Although this algorithm is based on empirical studies it allows you to get the right melody key.

3 Accompaniment generation algorithm

As accompaniment generation algorithm I choose the evolution algorithm. In my implementation there are classes that should be discussed:

3.1 Gene

Gene is simply the one chord of the accompaniment, but gene has an ability for mutation. If chord in gene is diminished, there will be no mutation at all, in other case: in Table 3 you can find probabilities of mutation.

Table 3: Probability of spesific mutation for gene

Probability	Mutation to
0.25	the first inverse
0.25	the second inverse
0.25	suspended 2
0.25	suspended 4

3.2 Chromosome

Each chromosome holds array of genes (chords of accompaniment), array has length that needed for accompaniment of the initial melody. Also, chromosome can evaluate genes that it has by using fitness function (described below). In addition, the chromosome can mutate its genes, this process is quite simple, going through genes and asking the gene to promote itself.

3.3 Generator

Generator is one of the essential classes. In fact, the whole implementation of evolution algorithm iterations is there. So, it has such abilities:

- Create ititial population of chromosomes of size that is specified by user
- Crossover two chromosome. This process is pretty random, after we get $chromosome_1$, $chromosome_2$, we pick prefix from $chromosome_1$ of random length and suffix of $chromosome_2$ and produce the new *chromosome* that will be of needed length of accompaniment.
- Get population fitness. We simply call fitness function of each chromosome.
- Produce next population based on previous population. As for me, It is the main function of this class. To construct the new population, it gets the sorted list of chromosome based on fitness function. Takes two best chromosomes and produces their children, and add mutated versions of them in new population array. Then again sort them all by fitness function. After all this steps we get sorted array in descending order of fitness value (so, best chromosomes will be at the beginning of the array), which length $3 \times$ population size (user-defined variable), that consists of previous iteration chromosomes and mutations of two best fitness-scored chromosomes. So, we make a slice of best chromosomes, which length is population size.
- Produce **n** iterations, where n is user-defined amount of generations, of the evolution algorithm.

3.4 Fitness function

Crucial function of evolution algorithm that estimate the chromosome, now I would like to introduce some rules, that was implemented. Disclaimer: I am not a musician at all :)
Function implemented as cumulative function, so, for some events I can reward, for others penalize.
For this function I divide song in parts that equals length of bar divided by 4 (beat), this is the length of one chord in generated accompaniment.

- Reward of 50 for each note in accompaniment that matches one of the note in original melody in this bit.
- Penalty of 20 if two sequential chords (genes) are equal.
- Penalty of 30 for every chord (gene) in accompaniment if it is sus2 or sus4.
- Reward of 50 if the first chord (gene) of accompaniment is tonic chord (triad).

4 Detected keys for input files

For us were given 4 files: `barbiegirl_mono.mid`, `input1.mid`, `input2.mid`, `input3.mid`. Now I would like to define the key of each melody:

Table 4: Detected keys

Name of file	Detected key
`barbiegirl_mono.mid`	`C#m`
`input1.mid`	`Dm`
`input2.mid`	`F`
`input3.mid`	`Em`

5 Source code

Python code, together with midi files, `requirements.txt` and this report sources are available in my GitHub repository, the link (just click on it).

References

[1] R. Hart, *Key-finding algorithm*, 19-Aug-2012. [Online]. Avalible: http://rnhart.net/articles/key-finding/. [Accessed: 26-Nov-2022].