Measurements are artificial perception

Marcus Bitzl - marcus@bitzl.io - https://www.bitzl.io

Introduction

Technology and Tools

General data processing: Python 3, Jupyter for interactive work, numpy, mido for MIDI generation. Matplotlib and seaborn for plotting (also visual structure of OCR).

Audio processing and mastering: Ableton Live suite, Ableton Collision physical modelling synthesizer.

Folder structure (Python project)

docs/
soundpiece/ ← *Python library*
data/
  src/ ← *source files as Downloaded*
  generated/ ← *generated files*
  manual/ ← m*anually edited files*
notebooks/ ← Jupyter notebooks
Pipenv
Pipenv.lock
README.md

Data preparation

After parsing the OCR data with lxml, the data structure has been reduced to lines containing words and saved as JSON file for later use.

Line:
  x: int
  y: int
  width: int
  height: int
  children: List

Word:
  x: int
  y: int
  width: int
  height: int
  content: str

Data Exploration

To explore the data, several approaches came to use:

- Visual analysis of line and word structure: Painting lines and alternating words in different colors (numpy, matplotlib) helped to gain a feeling for the visual structure.
- Statistical analysis: Counting characters, word lengths, characters per line,... helped to get an understanding of the amount and structure of the low level data.
- Analysis of line lengths, word coordinates etc. helped to gain an understanding of the finer visual structure as then represented in the melody.

Melody: Generate MIDI from structure and content

Normalization

The measurement tables are split into two layout columns at the end of the page. First we separated the OCR lines in the left and right part, then normalized both to start at x = 0 (each lines and words) and put them together as if it was one long table on the page.

Transforming coordinates to rhythm

The rhythm should reflect the physical layout of the table and the words in it. Therefore, we defined the line width for all rows as the span between the start of the leftmost row (x = 0) and the end of the rightmost row (not all rows start at zero as lines start in the OCR-Data where the first word starts).

All rows then where stringed up as if both tables where just one line, the start of each word was mapped to the start of a note, the end of the word to the note's end. To do so, after the end of the last word of a line and first word of the next line, an additional pause had to be added. The fact that the timings for MIDI events in a track in mido are relative to the previous event made this easier, as the pause could be treated as an offset for the words in the following line.

Transforming character to pitch

Transforming characters to a pitch had to be done in a semi-random way that offers enough variation to be interesting, but also enough repetition to keep the listener in. The vision for this soundpiece pictured a rather simple feeling for the melody as contrast to the complex voices, so each word should map to exactly one pitch.

As natural language does already have some statistical patterns and the tables also have certain repetitive structures, a simple mapping of character's ordinal values to a MIDI pitch value can be used. As statistical analysis showed that there is exactly one character with an

ordinal value larger than 200 (the frequent character „) and also that there are almost no characters with an ordinal value around 151, we chose to map this one to 151 to create a more homogeneous feeling (note that 127 is the maximum value for a midi pitch, but in this work we do not get too close to this value):

```
def word_to_note(word: str):
  x = sum(ord(c) for c in word) / len(word)
  if x > 8000:
    x = 151
  return int(x / 200 * 127)
```

The MIDI events were written to a single track MIDI file (format 1) and later importet into Ableton Live.

Creating voices using Google Cloud Speech API

The voices should evoke a mystic feeling of repetition and semi-chaotic variation like an incantation of technology, done by voices moving between almost natural and rather artificial. The language of the source material was German.

Therefore, a manually heavily cleaned up variant was created which also did replace all repetition marks by the word that should be repeated, and also all abbreviations have been replaced by their long forms. Then a python script was used to replace remaining special characters and numbers to be written as spoken (e.g. 1.23 → eins Komma zwei drei).

Using the Google Cloud Speech API, a list with all available German voices was determined. These include two voices using a simpler speech generation model, and four using the very realistic WaveNet AI system. For each voice a speech version of each line was generated (saved in a file with linenumber and sanitized version of the content as filename).

Creating the Soundpiece

At this point we continue using Ableton Live to create the soundpiece from the parts we generated so far:

The melody is imported as MIDI track with an Ableton Collision physical modelling synthesizer. Six audio tracks take samples from the voices (on track for each voice). One voice has been additionally filtered using a Flanger effect to create a more artificial feeling. The samples have all been manually selected from all the samples created in the step before.