NI–VMM – podobnost MIDI souborů

Bc. Vojtěch Kopecký

28. listopadu 2022

Obsah

1 Úvod **1**

2 Popis projektu **2**

3 Způsoby řešení **2**
3.1 Hledání hlavní melodie ve skladbě 2
3.2 Podobnostní algoritmy 3

4 Implementace **4**
4.1 Volba technologií a struktura projektu 4
4.2 Extrakce melodie 4

5 Příklad výstupu **5**

6 Měření **5**

7 Diskuze **5**

8 Závěr **5**

1 Úvod

Tato zpráva slouží jako dokumentace semestrálního projektu pro předmět NI–VMM. Zadáním projektu byla tvorba webové aplikace pro vyhledávání hudebních skladeb ve formátu MIDI na základě podobnosti melodií.

V této zprávě nejdříve blíže popíši vlastnosti projektu. Dále prozkoumám způsoby extrakce melodií a jejich porovnávání. V implementační části se zaměřím na zvolení vhodných technologií, práci s MIDI soubory (tím pádem také stručně popíši standard MIDI) a proces tvorby jak backendové, tak frontendové části aplikace. Předvedu příklady výstupů hotové aplikace a provedu základní měření rychlosti vyhledávání. Na závěr zhodnotím dosažené výsledky a navrhnu možná vylepšení aplikace.

2 Popis projektu

Zaměřil jsem se na klasické klavírní skladby – díky tomu, že se ve skladbě hraje jenom na jeden nástroj, je určení hlavní melodie značně jednodušší. Kromě toho jsou MIDI soubory pro klasickou hudbu na internetu velmi dostupné.

Vstupem vyhledávání je kratší soubor ve formátu MIDI (cca 4–40 po sobě jdoucích not), který obsahuje nějakou melodii. Aplikace se následně tuto melodii pokusí najít ve všech skladbách v databázi. Výstupem je seznam skladeb, které obsahují melodii nejvíce se podobající vstupní melodii. Tento seznam je setříděn od nejlepších (nejvíce podobných) výsledků po nejhorší.

Databáze obsahuje 295 klasických skladeb volně dostupných ze stránky `http://www.piano-midi.de/`.

Implementace projektu spočívala v několika krocích:

- Extrakce melodie z MIDI souborů a datová reprezentace melodie v databázi
- Porovnávání dvou melodií (určení jejich podobnosti)
- Vyhledávání nejpodobnější melodie v celé databázi
- Uživatelské rozhraní přizpůsobené pro práci s hudbou (např. umožňuje přehrávání, vizualizaci melodií…)

3 Způsoby řešení

3.1 Hledání hlavní melodie ve skladbě

Sekvence výšek

Za hlavní melodii považujeme sekvenci tónů, které jsou v daný čas skladby nejvyšší (tzv. výšky). Výhodou je jak lehká implementace extrakce, tak dostatečná přesnost – nejvyšší tóny většinou představují hlavní melodii, zvlášť v klavírních skladbách. Jsou však případy, kdy se hlavní melodie skrývá v basech (nejnižších tónech) a výšky jsou pouze komplementární.

Sekvence nejhlasitějších tónů

Dále můžeme za hlavní melodii považovat sekvenci tónů, které jsou v daný čas skladby nejhlasitější. Pro toto hledání nám už ale nestačí pouze notový zápis, ale nejlépe nějaký audio soubor – např. MP3, ve kterém bychom mohli analyzovat spektogram, nebo již zmíněný formát MIDI, ve kterém je hlasitost určitého tónu určena proměnnou velocity. Lze však usoudit, že i touto metodou bychom nedosáhli vysoce přesných výsledků – např. již kvůli tomu, že některé tóny mohou mít ve stejnou dobu stejnou hlasitost – poté bychom se museli dále rozhodovat podle výšky tónu či jiných vlastností.

Problémem je také fakt, že objektivně určit hlavní melodii nelze – záleží na subjektivní interpretaci dané skladby, což také vyžaduje určitou hudební znalost. I samotný uživatel, který melodii bude vyhledávat, si hlavní melodii může vyložit

jinak, než je obecně známo. U výše zmíněných metod proto nelze jednoduše ověřit, jak moc úspěšné jsou. Budeme tedy pracovat s domněnkou, že úspěšnost obou metod je dostatečná pro potřeby naší aplikace.

3.2 Podobnostní algoritmy

V obou metodách zmíněných v předchozí kapitole byly melodie reprezentovány sekvencí. To nám značně zjednodušuje situaci, jelikož vyhledávání podobných skladeb bude spočívat v porovnávání dvou sekvencí. Na tento problém existuje mnoho algoritmů, které jsou výpočetně rychlé a také jednoduché na implementaci.

Longest common subsequence (LCS)

LCS je problém nálezu nejdelší společné podposloupnosti dvou sekvencí. Příbuzným problémem je *Longest common substring* – neboli nález nejdelšího společného podslova. Slovo se od posloupnosti liší tím, že v obou sekvencích nesmí být přerušeno, zatímco společná posloupnost může přeskakovat prvky.

Při hledání podobných melodií je LCS užitečná – jedná se o jeden ze základních způsobů, jak určit podobnost sekvencí. LCS je navíc odolná proti malým změnám v melodii (např. jedna nota vložená navíc, či chybějící nota).

Dynamic time warping (DTW)

DTW je problém měření podobnosti dvou sekvencí, které se mohou lišit v rychlosti. Na první pohled tato metrika nemusí být pro hledání podobných melodií příliš užitečná, jelikož v extrahované melodii neuvádíme délku jednotlivých not, nýbrž pouze jejich tón. I přes to však DTW představuje sofistikovanější přístup k podobnosti melodií, než LCS, jelikož pracuje s odchylkou jednotlivých členů sekvencí a nezabývá se pouze jejich rovností – DTW tak může objevit podobnost dvou melodií, kde druhá melodie představuje první melodii posunutou např. o jeden půltón výše.

Jelikož mi jak LCS, tak DTW připadaly jako užitečné algoritmy k hledání podobnosti melodií, implementoval jsem je oba v mé aplikaci. Aplikace tedy bude sloužit také jako ukázka, v jakých případech si který algoritmus vede lépe. Efektivní algoritmy LCS a DTW využívají dynamického programování se složitostí $O(mn)$, kde m a n značí délky porovnávaných sekvencí.

4 Implementace

4.1 Volba technologií a struktura projektu

Aplikaci jsem rozdělil na backend a frontend. Backend bude zajišťovat veškerou „vědeckou“ práci, tedy extrakci melodie, správu databáze skladeb a samotné vyhledávání podobnosti. Backend tyto služby poskytne frontendu pomocí REST API. Na frontendu bude poté implementováno uživatelské rozhraní pro vyhledávání.

Pro backend se přirozeně nabízel jazyk Python, který disponuje mnoha knihovnami pro vědeckou práci, zároveň se dalo jednoduše experimentovat pomocí Jupyter notebooku, než všechno implementovat na ostro. Nejdříve jsem se však snažil najít Python knihovnu pro práci s MIDI soubory, abych extrakci nemusel implementovat ve frontendu (zde se nabízela Javascriptová knihovna *Tone.js*, která je schopna MIDI soubory konvertovat na JSON). Pro Python jsem našel knihovnu *Mido*, která k MIDI souborům přistupuje více low-level způsobem. I přes tento fakt jsem si vybral Python s tím, že knihovnu *Mido* vyzkouším. Jako web framework jsem zvolil *Django*, který se postará o samotné API a jakoukoliv práci s databází.

Pro frontend jsem zvolil Javascriptový framework *Vue 3* s grafickou knihovnou *Vuetify*.

4.2 Extrakce melodie

[1][2]

```python
from mido import MidiFile

# Load midi file
midi = MidiFile("../midi/piano_midi_de/bach/bach_846.mid")

# Print all messages in the second track
for message in midi.tracks[1]:
    print(message)
```

```
MetaMessage('midi_port', port=0, time=0)
MetaMessage('track_name', name='Piano right', time=0)
program_change channel=0 program=0 time=0
control_change channel=0 control=7 value=100 time=0
control_change channel=0 control=10 value=64 time=0
control_change channel=0 control=91 value=127 time=0
MetaMessage('text', text='bdca426d104a26ac9dcb070447587523', time=0)
note_on channel=0 note=67 velocity=56 time=241
note_on channel=0 note=67 velocity=0 time=120
note_on channel=0 note=72 velocity=60 time=0
note_on channel=0 note=72 velocity=0 time=120
note_on channel=0 note=76 velocity=63 time=0
```

```
note_on channel=0 note=76 velocity=0 time=108
...
note_on channel=0 note=72 velocity=0 time=0
MetaMessage('end_of_track', time=0)
```

5 Příklad výstupu

We did some experiments ...

6 Měření

From our experiments we can conclude that ...

7 Diskuze

8 Závěr

Odkazy

1. *Mido - MIDI Objects for Python* [online]. [cit. 2022-11-27]. Dostupné z: `https://mido.readthedocs.io/en/latest/`.
2. *Standard MIDI-File Format Spec. 1.1, updated* [online]. [cit. 2022-11-27]. Dostupné z: `http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html`.