TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Anna Talas

Muusikateose vormianalüüs loomulikus eesti keeles

Bakalaureusetöö (9 EAP)

Juhendaja: Sven Aller, MSc

Tartu 2022

Muusikateose automaatne vormianalüüs loomulikus eesti keeles

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus, mis võimaldab kasutajal vabalt valitud helifailina esitatud muusikateose vormi analüüsida. Töö raames antakse ülevaade olemasolevatest muusika kirjeldamise rakendustest ning analüüsitakse nende tugevusi ja puudujääke. Järgnevalt antakse ülevaade erinevatest võimalustest muusikateose struktuuri tuvastamiseks ning töötatakse välja ka algoritm automaatseks vormianalüüsiks, mille põhjal ennustatakse teose vormi valitud lihtvormide hulgast. Lisaks sellele kirjeldatakse valminud rakenduse „Vormileidja“ ülesehitust ja kasutusvõimalusi. Viimaks pakutakse edasiarendamise võimalusi nii vormianalüüsi kui veebirakenduse parendamiseks.

Võtmesõnad:

Muusika, kirjeldus, vormianalüüs, automaatne analüüs, muusikateooria, loomulik eesti keel

CERCS: P175 Informaatika, süsteemiteooria

Automatic form analysis of music in natural Estonian language

Abstract:

The aim of this Bachelor’s thesis is to create a web application which allows the user to analyse the structural features of any given musical piece which has been provided as an audio file. The paper gives an overview of existing applications which provide information about musical pieces and analyses their strengths and shortcomings. An overview of different approaches to music structure analysis is given and an algorithm for feature based segmentation is constructed, which is also used to predict the form of a musical piece from a list of selected sectional forms. In addition the structure of the web application „Vormileidja“, which was built as a part of the thesis, is described as well as its functionality. Lastly ideas for future works are provided in order to improve form analysis as well as the web application.

Keywords:

Automatic music description, form analysis, automatic analysis, music theory, Estonian language

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Muusika kuulamine on suur osa paljude inimeste igapäevasest elust ning muusikaõpetus on Eestis ka osa riiklikust õppekavast. Tänapäeval on järjest rohkem tekkinud veebilehti, mis muudavad lihtsamaks vajaliku info leidmise heliteoste või artistide kohta, kuid enamus veebilehti pakuvad vaid infot suurtest andmebaasidest, mis keskenduvad peamiselt inglisekeelsele levimuusikale.

Seevastu Eestis nii üldhariduskoolides kui ka muusikakoolides keskendub muusikaharidus peamiselt klassikalisele muusikale, mille struktuur on enamasti keerukam kui levimuusikas ning erinevaid muusikalisi vorme on väga palju. Muusika iseseisev analüüsimine, eriti just vormianalüüs, nõuab aga eelteadmisi ja aega ning puudub võimalus oma analüüsi tulemusi kontrollida. Samuti teeb vormianalüüsi raskeks asjaolu, et ka muusikateooriat õppinud inimeste arvamused ei pruugi alati ühtida. Lõputöö kirjutamise ajal ei leidu autorile teadaolevalt ühtegi rakendust, mis kasutaja antud helifaili põhjal suudaks teostada heliteose vormianalüüsi ning selle põhjal ennustada vormi.

Antud bakalaureusetöö eesmärk on töötada välja algoritm, mis suudab tuvastada heliteose struktuuri ning seejärel selle põhjal ennustada teose vormi valitud lihtvormide hulgast nagu kolmeosaline lihtvorm, rondo, variatsioonivorm jt. Seejärel luua veebirakendus, mis suudab etteantud helisalvestuse alusel tagastaks kasutajale tulemuse sidusas eesti keeles, mis annaks infot nii teose struktuuri kui ka muude näitajate nagu tempo ja helistiku kohta. Selline rakendus võimaldab kasutajal olenemata muusikalisest taustast helifaili põhjal tuvastada teose struktuuri ning kuna rakendus on eesti keeles, siis on see kasutatav ka noorematele kooliõpilastele, kes ei pruugi veel võõrkeeli osata.

Lõputöö raames analüüsitakse kõigepealt olemasolevaid rakendusi muusikateoste kirjeldamiseks ning nende tugevusi ja puudujääke. Seejärel uuritakse erinevaid viise tuvastada heliteose puhul erinevaid parameetreid, keskendudes peamiselt just muusikateose vormi tuvastamise viisidele. Viimaks kirjeldatakse lõpptulemusena valminud veebilehte, kuhu kasutajal on võimalik üles laadida helifail, mille töötluse tulemusena veebileht visualiseerib heliteose vormi ning annab selle kohta ka lisainfot.

1. Olemasolevad rakendused muusikateose kirjeldamiseks

See peatükk pakub ülevaadet olemasolevatest muusikateose kirjeldamise rakendustest ning analüüsib olemasolevate lahenduste tugevusi ja nõrkusi.

1.1 GetSongKey ja GetSongBpm

GetSongKey [1] on veebirakendus, mille järgi saab leida heliteose helistiku. Rakendus pakub võimalust otsida andmebaasis leiduvate lugude kohta infot, mille kohta lisaks helistikule pakub rakendus infot ka loo positiivsuse, tantsitavuse, energia, akustilisuse ja reipuse kohta. Samuti pakub rakendus otsitud loo juures sarnaseid lugusid ning lugusid on võimalik otsida ka helistiku järgi. Rakendus pakub ka võimalust laadida üles oma helifail, mille peale rakendus tagastab selle loo helistiku – küll aga kui on tegu looga, kus esineb mitu helistikku, siis tagastatakse vaid kõige dominantsem. GetSongKey on peamiselt suunatud diskoritele [1], et leida lugusid, mis omavahel hästi kokku sobiks. Samade autorite pool on loodud ka rakendus GetSongBpm [2], mis võimaldab tuvastada muusika tempot, kasutades ühikuna *beats per minute* ehk lööki minutis. Sarnaselt rakendusele GetSongKey pakub rakendus andmebaasis leiduvate heliteoste kohta lisainfot.

1.2 Chosic

Chosic on veebirakendus, mis võimaldab tuvastada muusikateose žanrit [3]. Lugu saab otsida nii nime, artisti, žanri kui ka lingi kaudu. Tulemusena kuvatakse heliteost iseloomustavad žanrid märksõnadena, mis pärinevad Spotifyst, lehelt LastFM ning artisti Wikipedia lehelt. Lisaks sellele kuvatakse ka muid omadusi nagu loo kestvus ja tempo, kuid ka subjektiivsemad näitajaid nagu loo rõõmsus, tantsitavus ja energia. Sarnaselt eelnevatele rakendustele on ka Chosic puhul kuvatav info pärit erinevatest andmebaasidest, kuid antud lehel puudub ka võimalus üles laadida oma helifaili analüüsimiseks. Veebilehe peamiseks eesmärgiks tundub olevat erinevate žanrite ja artistide avastamine, põhinedes soovitusalgoritmil, mis võimaldab peale loo otsimist kuvada ka sarnaseid lugusid ja artiste.

1.3 ChordAI

ChordAI [4] on telefonirakendus, mis helifaili põhjal tuvastab heliteoses kasutatud akordid ning kaardistab nende asukohad helifailil. ChordAI võimaldab nii üles laadida oma helifaili, otsida andmebaasis olemasolevaid lugusid kui saada ka mängitavaid akorde koheselt, kasu-

tades telefoni mikrofoni. Rakendus on eelkõige suunatud kitarrimängijatele lugude saatepartiide õppimiseks ning vaikimisi kujutab akorde akordikujudena kitarri sõrmlaual. Tasulise versiooni korral on võimalik analüüsitud teose akorde salvestada ka PDF-faili kujul.

1.4 Muusik

Muusik on Maria Pibilota Murumaa poolt lõputöö raames loodud rakendus [5], mis helifaili põhjal annab eesti keeles ülevaate heliteose põhilistest omadustest nagu tempo, helistik ja heliteose kulminatsiooni ajahetk. Samuti pakub rakendus erinevaid märksõnu, millega heliteost saab kirjeldada, ning hinnangu heliteose energilisusele ning tantsitavusele. Leitud info edastab rakendus loomulikus keeles sidusa tekstina. Lõputöö kirjutamise ajal on rakendus jooksutatav vaid lokaalses arvutis.

1.5 Muusikakirjeldus

Villem Tõnissoni magistritöö raames loodud rakendus [6] annab samuti helifaili põhjal ülevaate heliteose põhilistest omadustest. Minimalistliku disainiga rakenduses saab kasutaja valida, mille kohta ta soovib informatsiooni saada, järgnevatest omadustest: tempo, helistik, harmoonia, vorm, žanr ja instrumendid. Enamus omadustest leitakse masinõppe mudelite kaudu ning žanritest pakutakse infot levimuusika žanrite kohta nagu popmuusika, elektrooniline, hip-hop, rokk ning lisaks sellele tuvastab ka instrumentaal- ning eksperimentaalmuusika. Võrreldes eelnevalt kirjeldatud rakendusega Muusik [5] teostab antud rakendus küll põhjalikuma analüüsi, kuid loomulikus keeles esitatud väljund on väga minimalistlik.

1.6 Analüüs

Kuigi erinevaid rakendusi muusikateoste kohta info saamiseks on olemas mitmeid, siis keskendub enamus neist annavad heliteose kohta vaid andmebaasides leiduvat infot, mis tähendab, et vähem tuntud või teistes keeltes olevate teoste kohta pole enamasti võimalik iseloomustust saada. Seevastu rakendused, mis töötlevad vaid helifaili ennast, pakuvad enamasti väga piiratud infot, näiteks ainult tempot ja helistikku ning samuti puudub loomulikus keeles väljund. Siinkohal on erandiks 2021. aastal Maria Pibilota Murumaa ja Villem Tõnissoni poolt lõputööde raames loodud rakendused. Samas on mõlema rakenduse negatiivseks küljeks see, et need on vähemalt hetkeseisus vaid lokaalselt jooksutatavad rakendused, mis eeldavad vastava tarkvara eelnevat paigaldust.

Kuigi klassikaline muusika on muutumas populaarsemaks [7], keskenduvad enamus rakendused vaid levimuusika žanritele ning lõputöö käigus uuritud rakenduste hulgas ei leitud selliseid, mis suudaks detailselt analüüsida klassikalise muusika teoseid, kus helistik ja tempo pole tihti kogu loo jooksul üheselt määratud. Samuti pole antud rakendused võimelised eristama erinevaid klassikalise muusika vorme.

Käesoleva bakalaureusetöö eesmärgiks on luua eestikeelne rakendus, mis helifaili põhjal tagastab muusika vormianalüüsi. Eesmärgiks on heliteose puhul lisaks tuvastada ka põhilised näitajad nagu tempo ja helistik, kuid lisaks sellele keskendub lõputöö klassikalise muusika poole pealt teose vormi ära tundmisele, milleks ükski leitud rakendustest hetkeseisus võimeline pole. Tulemusena tagastatakse kasutajale leitud omadused sidusa tekstina eesti keeles ning lisaks vormi kujutav joonis. Samuti on eesmärgiks teha rakendus kasutajale lihtsasti kättesaadavaks veebilehe kujul.

2. Kasutatud tehnoloogilised vahendid

Käesoleva peatükiga antakse lühike ülevaade tehnoloogilistest vahenditest, mida kasutati rakenduse loomiseks. Rakenduse tagaliides (ingl *back-end*) on loodud, kasutades programmeerimiskeelt Python ning selle teeke, kasutajaliidese poolel on kasutatud Javascripti ning märgistuskeeli HTML ja CSS.

2.1 Python

Rakenduse loomiseks on kasutusel Pythoni versioon 3.8 ning erinevad teegid, mis võimaldavad heliteose töötlemist. Programmeerimiskeel Python sai valitud just heli- ja andmetöötluseks olemasolevate teekide rohkuse tõttu.

2.1.1 Librosa

Pythoni teek librosa [8] on mõeldud audio analüüsimiseks. Librosa teek pakub funktsioone helifailist põhiliste tunniste leidmiseks. Samuti on võimalik antud teegi abil töödelda helifaile helisageduste ja helitugevuse põhjal, luua ja töödelda sarnasusmaatrikseid ning segmenteerida heliteoseid sarnasuse põhjal. Lisaks sellele sisaldab librosa teek funktsioone vastavate spektrogrammide ja maatriksite mugavaks visualiseerimiseks.

Teegi looja Brian McFee sõnul [9] sai librosa teek loodud eesmärgiga võimaldada muusika analüüsimist programmeerimiskeeles Python teadlastele, kes seni kasutasid peamiselt keeli nagu MATLAB või C++. McFee peab teegi disaini puhul ka väga oluliseks modulaarsust ja paindlikkust, mis laseks kasutajatel mugavalt kasutada teegi funktsioone ning samal ajal soovi korral eksperimenteerida läbi erinevate komponentide täiustamise. Käesoleva lõputöö raames sai librosa teek valitud heliteoste analüüsimiseks just tänu teegi modulaarsusele ning mugavatele visualiseerimist pakkuvatele funktsioonidele.

2.1.2 Madmom

Madmom [10] on helisignaali töötlemise teek, mis keskendub peamiselt heliteostest informatsiooni eraldamisele. Teegi loojate eesmärk on pakkuda lihtsat viisi töödelda helifaile ilma, et peaks sõltuma paljudest erinevatest teekidest või tarkvarast. Madmom pakub võimalusi nii helisignaali madalal tasemel töötlemiseks kui ka erinevaid masinõppe mudeleid heliteose tunnuste tuvastamiseks. Antud lõputöös on kasutatud Madmom teeki just tänu võimalusele rakendada eeltreenitud masinõppe mudeleid helisignaali töötlemiseks.

2.1.3 SciPy

SciPy [11] on teadusarvutuse teek, mis pakub erinevaid algoritme optimeerimiseks, võrrandisüsteemide lahendamiseks, pilditöötluseks, signaalitöötluseks kui ka statistikaks. Teegil on olemas ka põhjalik dokumentatsioon koos koodinäidetega. Teek loodi aastal 2001 eesmärgiga luua keskkond teaduslikuks andmeanalüüsiks ning on nüüdseks üks enimkasutatud Pythoni teeke teaduslike algoritmide loomisel [12]. Peatükis 3 kirjeldatud vormianalüüsis kasutatakse SciPy teeki nii filtreerimiseks kui ka klasterdamiseks.

2.2 Flask

Flask [13] on programmeerimiskeelel Python põhinev raamistik, mille abil on võimalik lihtsalt luua veebirakendusi. Flaski eesmärk on olla nii-öelda mikroraamistik, mis lubab mugavalt implementeerida põhifunktsionaalsust, kuid laseb arendajal vajadusel lisada oma soovi järgi laiendusi lisafunktsionaalsustekst, näiteks puudub raamistikul vaikimisi andmebaasi integratsioon.

Vijay Singh võrdleb artiklis „Flask vs Django in 2022: Which Framework to Choose?“ [14] Pythoni populaarsemaid veebiraamistikke Flask ja Django. Singh jõuab järeldusele, et Flask on parem väiksemate rakenduste loomiseks ning parem valik algajale veebiarendajale, kuna on lihtne omandada ning väga algelise veebilehe loomiseks on vaja vaid paari rida koodi. Seevastu Django õppimine on ajamahukam ning selle raamistiku põhieelised tulevad välja alles suuremate rakendustega, millel on vaja rohkem funktsionaalsust. Kuna antud lõputöö raames loodav rakendus on üsna minimalistlik, osutus valitud raamistikuks just Flask.

2.3 Heroku

Heroku [15] on veebirakendus, mis pakub pilvepõhist teenus veebilehtede majutamiseks. Lõputöö raames loodud rakenduse majutamiseks valisin Heroku peamiselt just selle pärast, et Heroku toetab Pythoni kasutust ning kuigi enamus majutusplatvorme eeldavad tasusid serverite kasutamise eest, siis Herokus on väiksemate rakenduste majutus tasuta.

2.4 Docker

Docker [16] on virtuaalsete konteinerite loomise ja haldamise tehnoloogia, mis lihtsustab arendusprotsessi vähendades konfigureerimisvajadust rakenduse erinevatel seadmetel jooksutamiseks. Lõputöös loodud rakenduse mugavaks üleslaadimiseks Heroku serverisse on kasutatud Dockeriga loodud tõmmist.

3. Vormi analüüsimine

Kerri Kotta muusikateooria õpik [17] defineerib muusikateose vormi kui muusika ajalise liigendusstruktuuri. Lõputöö kirjutamise ajal pole olemas ühtegi teeki, mis pakuks funktsiooni heliteose vormi jaotuseks. Antud peatükk käsitleb erinevaid viise helifaili liigendada muusikas korduvate teemade järgi ja käesolevas lõputöös väljatöötatud lahendust koos testimiseks loodud rakendusega ning selle põhjal vormi nimetuse ennustamist.

3.1 Olemasolevad uurimustööd

Enne algoritmi loomise alustamist sai lõputöö raames uuritud erinevaid olemasolevaid lähenemisi heliteose vormianalüüsiks. Uuritud lähenemised saab jagada kaheks: masinõppel põhinev ning algoritmiline. Järgnevalt on kirjeldatud kolme erinevat lähenemist heliteose struktuuri leidmiseks.

Pychorus [18] on Vivek Jayaram poolt loodud Pythoni teek, mis võimaldab leida heliteose refrääni. Loodud teek põhineb Masataka Goto uurimustööl „A chorus section detection method for musical audio signals and its application to a music listening station“ [19]. Teegi abil saab andes ette helifaili ning otsitava refrääni pikkuse leida ja eraldi faili salvestada parima leitud tulemuse. Pychorus teek kasutab refrääni tuvastamiseks ajavahemaatrikseid (ingl *time-lag matrix*), millelt üritab tuvastada horisontaalseid jooni. Iga horisontaalne joon maatriksil kujutab korduvat lõiku heliteoses ning leides iga lõiguga paralleelsete lõikude arvu on võimalik tuvastada mitu korda antud teema muusikapalas kõlab. Samas ka autori enda sõnul ei tööta teek muusikateoste korral, mille puhul tempo võib kõikuda, näiteks klassikaline muusika või ka vanem levimuusika, kuna sel juhul ei pruugi korduvaid teemasid kujutavad jooned olla päris horisontaalsed [20]. Samuti on teek võimeline leidma vaid kõige populaarsemat teemat ning tagastab ühe peateema esinemiskohtadest teoses.

Librosa teegi näidiste hulgas on Laplace’i segmentatsioonil põhinev heliteose liigendamine [21], mille aluseks on McFee ja Ellis uurimustöö „Analyzing Song Structure with Spectral Clustering“ [22]. Heliteose liigendamisel luuakse heliteose sarnasusmaatriksist (ingl *self-similarity matrix*) normaliseeritud Laplace’i maatriks, mis seejärel jaotatakse eelnevalt määratletud arvuks erinevateks struktuurikomponentideks. Seejärel kasutatakse scikit-learn teegist funktsiooni KMeans, et ennustada eelnevalt määratud arvu erinevate teemade jaotust heliteoses.

Meinard Mülleri raamatu „Fundamentals of Music Processing – Using Python and Jupyter Notebooks“ [23] põhjal loodud Jupyter Notebook koodinäidete hulgas keskendub 4. peatükk muusika struktuuri analüüsimisele [24]. Läbi erinevate koodinäidete on välja toodud erinevad algoritmilised lähenemised helifaili töötlemisele kasutades nii sarnasusmaatrikseid kui ajavahemaatrikseid ning erinevaid silumis ning filtreerimisalgoritme. Eelkõige erineb pakutud lähenemisviis eelnevatest selle poolest et kasutatakse uudsusfunktsiooni (ingl *novelty function*), mis leiab erinevused ajahetkede vahel ning funktsiooni lokaalsete maksimumide põhjal saab ennustada heliteoses teemade vahetuskohti. Eelnevalt kirjeldatud lähenemine põhineb Serrà, Müller jt uurimustööl „Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity“ [25].

Vaadeldud lähenemistest on masinõppe lahendus suure tõenäosusega lihtsaim implementeerida, kuna vastavad mudelid ning funktsioonid, mis teevad põhilise töö, on teekides nagu *librosa* juba olemas koos näidistega. Sellise lähenemise suurimaks nõrkuseks on aga, et üldjuhul on vaja mudelile ennustamisel ette anda info selle kohta, kui mitmeks erinevaks osaks teost soovitakse jaotada, mida üldjuhul ei ole võimalik kindlalt öelda enne helifaili eelnevat analüüsimist. Algoritmilistest lähenemistest on ajavahemaatriksitelt horisontaalsete lõikude tuvastamine intuitiivselt lihtsasti arusaadav, kuid peaaegu kunagi ei saa garanteerida kõigi joonte korrektset tuvastamist. Pychrous näitel vaid refrääni tuvastamise vaatepunktist pole oluline kõigi joonte tuvastamine, kuna piisab paarist, et leida kõige enam korduv teema. Kogu heliteose struktuuri genereerimise puhul võib aga selline informatsioonikadu mõjutada lõpptulemust väga tugevalt. Selle tõttu on antud lõputöö raames vormianalüüsile lähenetud algoritmiliselt ning eelkõige näiteks võetud Serrà, Müller jt uurimustöö [25].

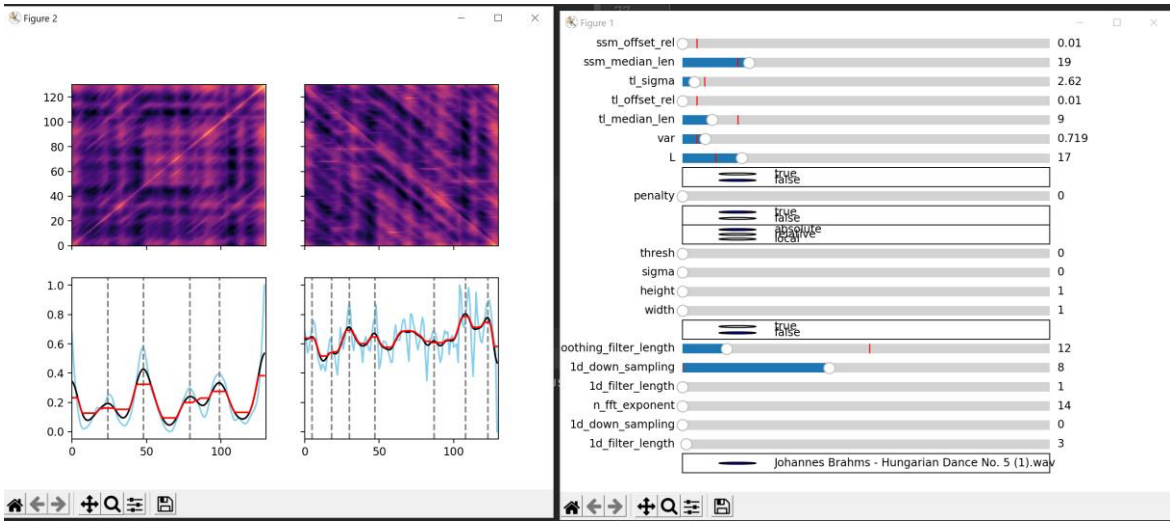
3.2 Rakendus testimiseks

Peale uurimustööde analüüsimist sai paika pandud algne plaan töötluseks. Kuna vaid uurimustööde lugemise põhjal pole võimalik ennustada, milline lähenemine on parim, siis mitmete töötlussammude puhul sai testitud erinevaid variante ning neid omavahel võrreldud.

Selleks, et teosest vorm kätte saada, on vaja läbida mitmeid andmeanalüüsi etappe, millest suur osa on parametriseeritud ning nende parameetrite mõju tulemusele võib olla etteaimamatu. Käsitsi testimine lähtekoodis parameetreid muutes on aga väga aeganõudev tegevus.

Selleks, et nendes paremini orienteeruda, sai loodud lisarakendus *paramviz*, mis on leitav ka GitHubi repositooriumis[1].

Interaktiivne rakendus on loodud Pythoni teegi *matplotlib* abil ning see võimaldab muuta erinevaid töötlusel kasutatud funktsioonide parameetreid ja visuaalselt näha erinevate nende efekti vormi töötlusel. Joonis 1 on näha kuvatõmmis rakendusest, kus vasakul pool on näha töötluse tulemusena loodud joonised ning paremal eraldi akent parameetrite muutmiseks. Vasakpoolses aknas on näha ülemisel real vastavalt sarnasusmaatriks ning ajavahemaatriks ning alumisel real vastavate maatriksite põhjal loodud uudsuse funktsioon ning vertikaalsed jooned näitamaks ennustatavat osade jaotust heliteoses.



Joonis 1. Testimiseks loodud rakendus.

3.3 Algoritmi kirjeldus

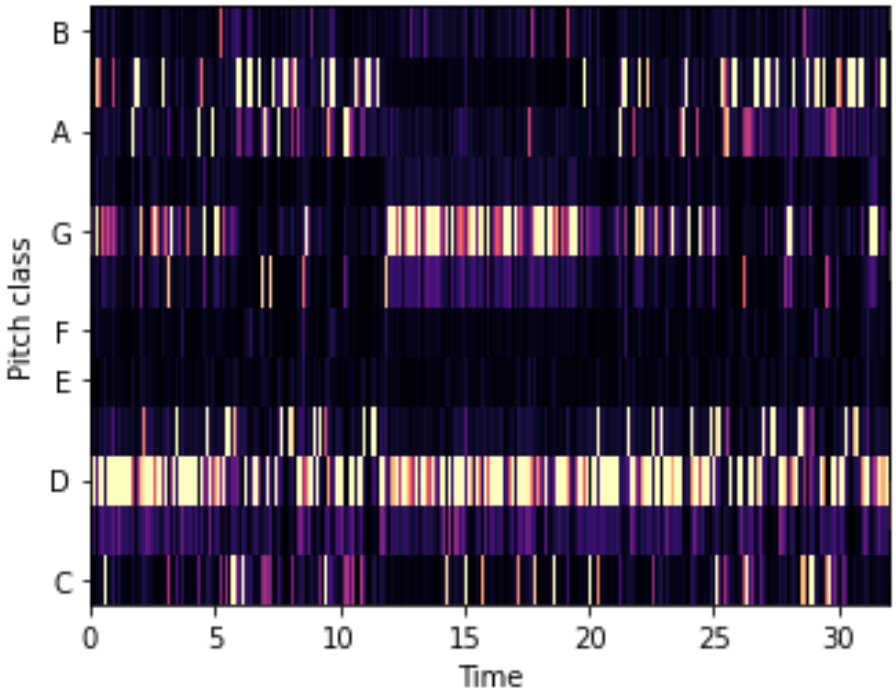
Erinevate parameetrite ning lähenemiste testimise tulemusena sai välja töötatud järgnev lahendus, mida on kasutatud ka loodud veebirakenduseks. Antud peatükis kasutatud näitlikustavad joonised on Johannes Brahmsi "Ungari tants nr. 5" helifaili analüüsi põhjal loodud.

Esmalt koos heliteose sisse laadimisega leiame ka selle diskreetimissageduse (ingl *sample rate*). Testimise käigus selgus, et üheks suuremaks probleemiks töötlusel on liigne müra, mida esineb rohkem just kõrge diskreetimissagedusega helifailide puhul, seega vajadusel

[1] https://github.com/mannapuder/bakatoo/tree/main/paramviz_app

vähendame helisignaali diskreetimissagedust. See aitab vähendada liigseid detaile, mis võivad tõlgenduda lihtsalt müraks ning samuti vähendab see järgmistele sammudele kuluvat aega. Samuti eemaldame helifaili algusest ja lõpust vaikust sisaldavad osad *librosa.effects.trim* [26] funktsiooni abil, kuna tegu on vormianalüüsi vaatepunktist ebaolulise infoga.

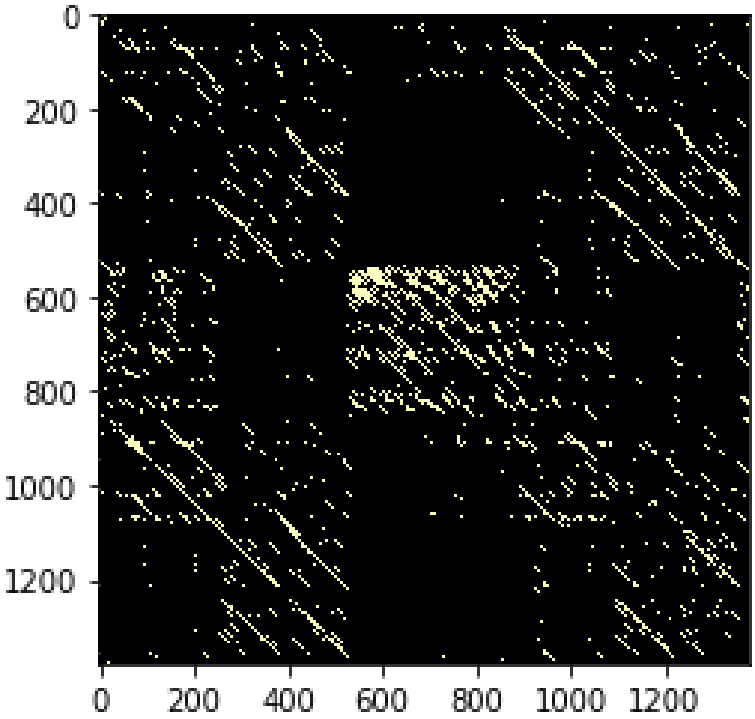
Helisignaali põhjal loome esmalt kromaatiliste tunnuste maatriksi mida on näha ka joonisel 2. Maatriksi loomiseks eraldame esmalt helisignaalist perkussioonid, jättes alles vaid harmoonilised komponendid, kasutades librosa teegi funktsiooni *librosa.effects.harmonic* [27]. Selle alusel koostame helisageduste spektrogrammi kasutades konstantse Q teisendust (ingl *Constant-Q transformation*), milles jaotame spektrogrammi sisu kromaatiliste tunnuste maatriksi helikõrguste klassideks (ingl *pitch class*) ning seda kõike läbi *librosa.effects.chroma_cqt* funktsiooni [28]. Viimaks vähendame veel maatriksist müra *librosa.decompose.nn_filter* [29] abil, mis asendab iga andmepunkti tema lähimate naabrite agregaadiga.



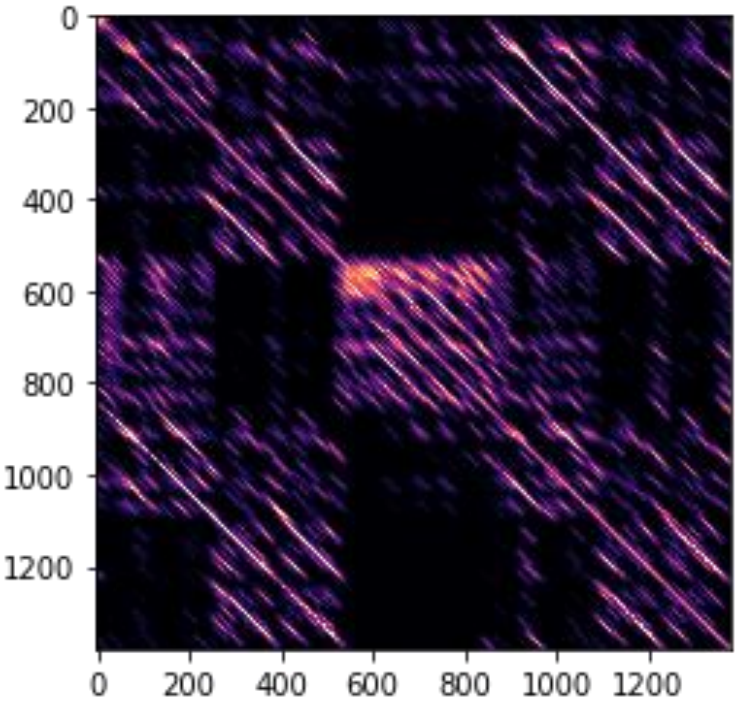
Joonis 2. Helifaili põhjal loodud kromaatiliste tunnuste maatriks.

Töödeldud kromaatiliste tunnuste maatriksi põhjal loome omakorda esmalt Joonis 3 näha oleva sarnasusmaatriksi funktsiooni *librosa.segment.recurrence_matrix* [30] abil. Kuna saadud maatriks on üsna granulaarne ning maatriks sisaldab endiselt suurt hulka müra, siis teostame maatriksi peal silumist *librosa.segment.path_enhance* [31] funktsiooniga, mille

tulemus on nähtav Joonis 4. Kahe joonise võrdlemisel on näha, et peale silumist on selgemalt välja tulnud diagonaalsed jooned mis tähistavad korduvaid teemasid heliteoses.

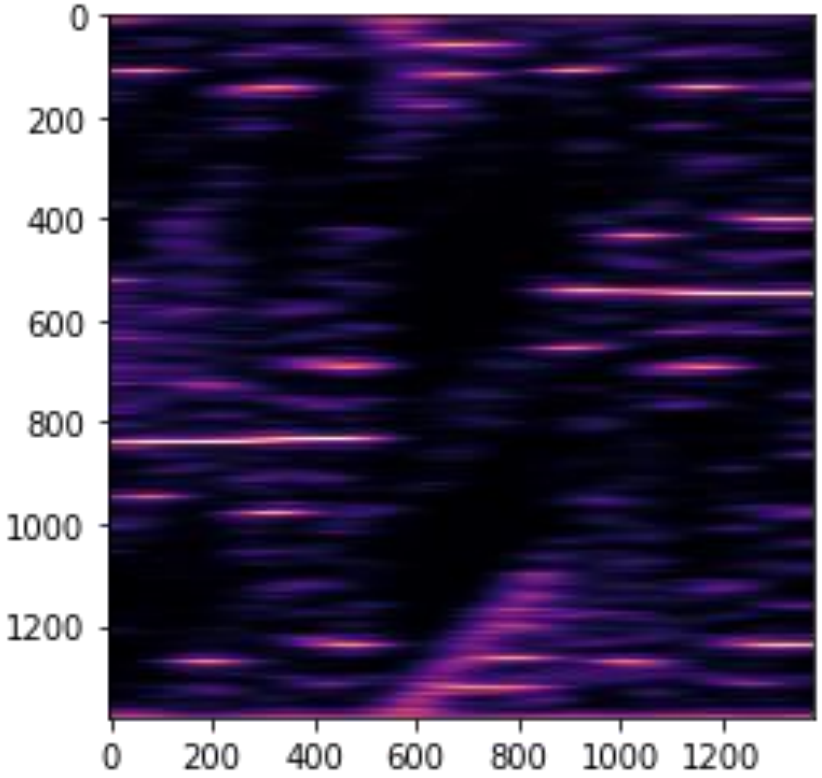


Joonis 3. Heliteose sarnasusmaatriks.



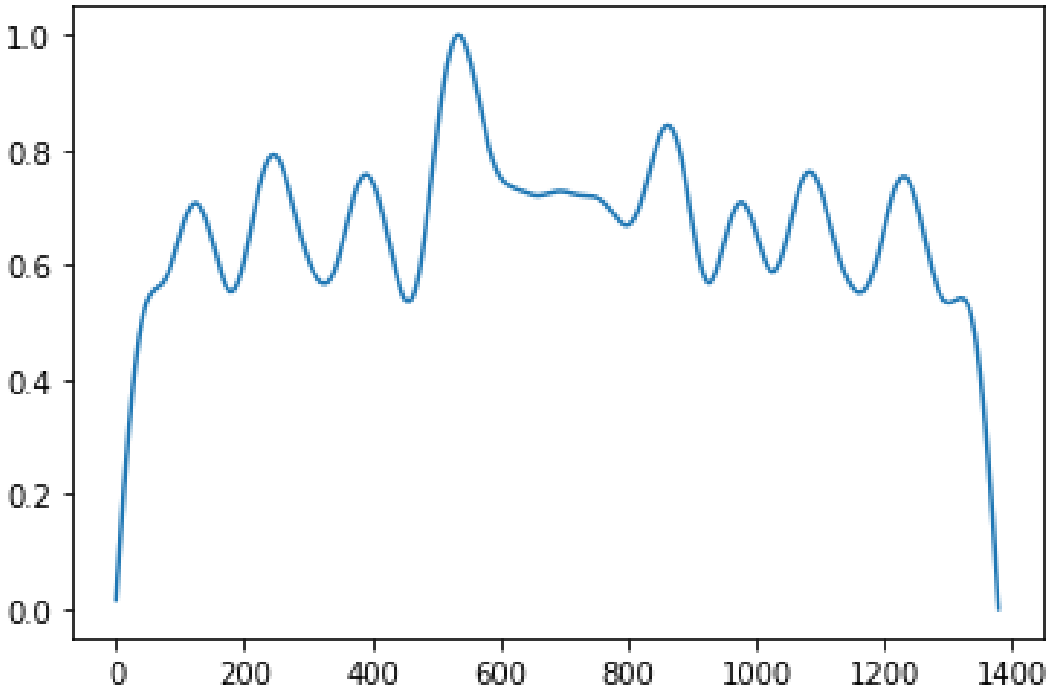
Joonis 4. Silutud sarnasusmaatriks.

Sarnasusmaatriksi põhjal loome edasiseks töötluseks vajaliku ajavahemaatriksi *librosa.segment.recurrence_to_lag* [32] funktsiooniga. Testimise jooksul selgus, et täpsema tulemuse saavutamiseks tasub ka ajavahemaatriksit siluda müra vähendamiseks ning selleks on kasutatud Gauss'i filtrit SciPy teegi funktsiooni *scipy.ndimage.gaussian_filter1d* [33]. Joonis 5 on näha töötluse tulemus peale silumist.



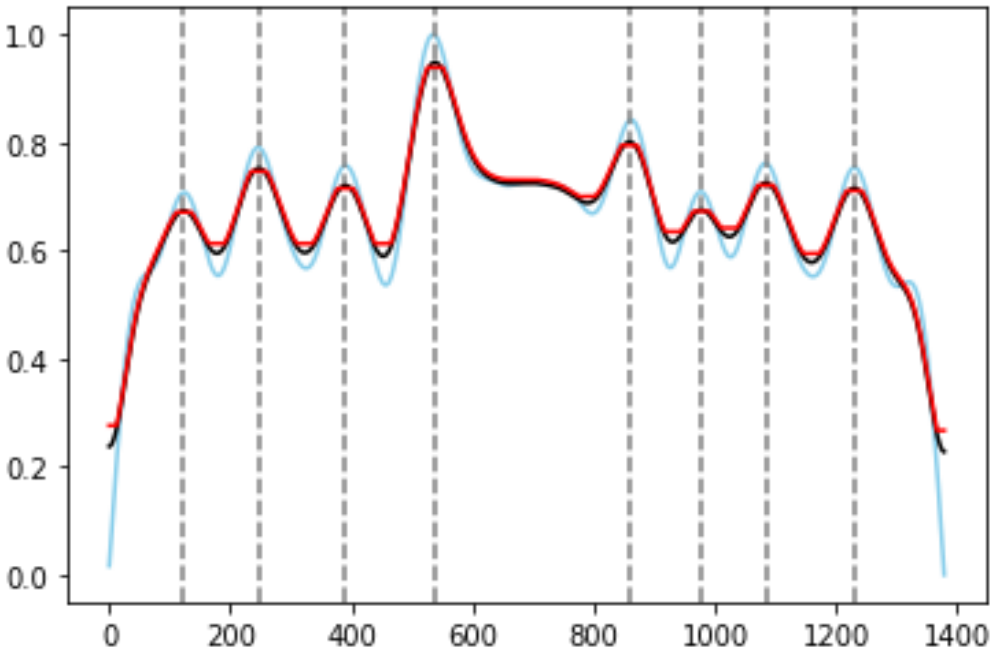
Joonis 5. Silutud ajavahemaatriks.

Selle jaoks, et leida töödeldud maatriksi põhjal heliteose struktuuri, on vaja eelnevalt tuvastada need ajahetked heliteoses, kus üks teema lõppeb ja teine algab. Antud lahenduse puhul kasutame nende tuvastamiseks uudsusfunktsiooni, mille tulemus on kujutatud Joonis 6. Funktsioon on arvutatud silutud ajavahemaatriksi põhjal, kasutades NumPy teegi funktsiooni *linalg.norm*. Selle uudsusfunktsiooni algoritm on pärit eelnevalt mainitud "Fundamentals of Music Processing" raamatu põhjal loodud Jupyter Notebookide kogumiku peatükist "Structure Feature" [34].



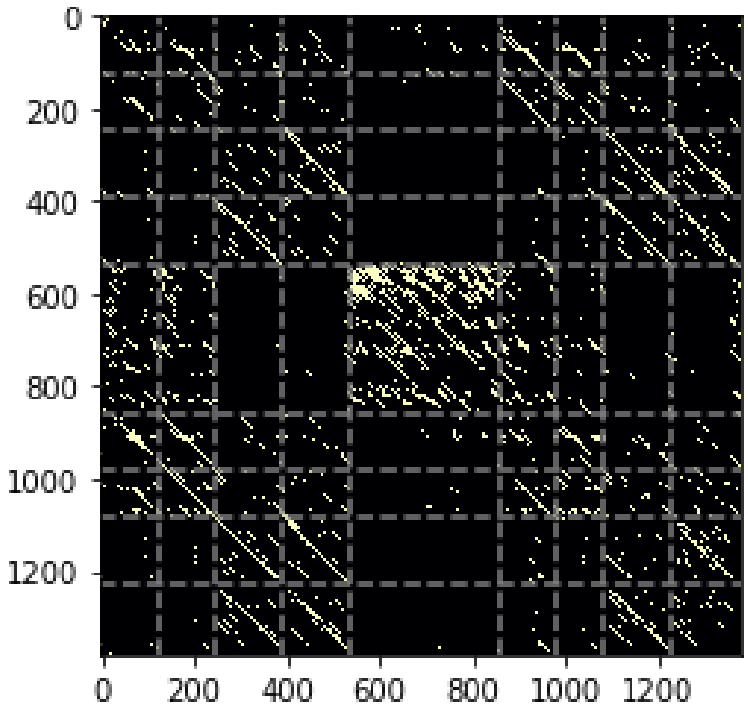
Joonis 6. Uudsusfunktsioon.

Antud uudsusfunktsiooni puhul tähistavad lokaalsed maksimumid ajahetki teoses, kus toimub kõige suurem muusikalise materjali muutus võrreldes eelnevaga ning seetõttu on need head kandidaadid uue teema alguskohaks. Kuigi antud näite puhul on lokaalsed maksimumid üsna selgelt eristunud, siis testimise käigus selgus, et see pole nii kõigi heliteoste puhul ning tihti tekivad funktsioonidesse väiksemad lokaalsed maksimumid, mille puhul on tegu valepositiivsetega. Nende vältimiseks rakendame uudsusfunktsioonil kõigepealt sarnaselt ajavahemaatriksile filtreerimiseks funktsiooni *scipy.ndimage.gaussian_filter1d* [33] ning seejärel lokaalset silumist mediaanfiltri *scipy.ndimage.median_filter* [35] abil. Seejärel valime heliteose struktuuri poolituskohtadeks need lokaalsed maksimumid, mis on suuremad kui silutud funktsioon. Valitud tipud on märgitud Joonis 7 vertikaalsete joontena ning lisaks on näha uudsusfunktsioon sinise, esmase filtreerimise tulemus musta ning lokaalselt silutud funktsioon punase joonega.



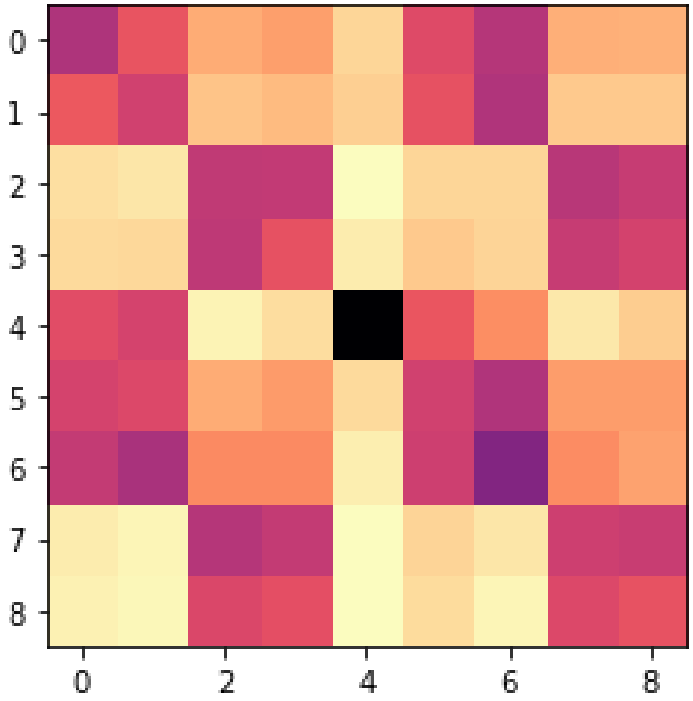
Joonis 7. Töödeldud uudsusfunktsioon koos lokaalsete maksimumidega.

Kandes leitud lokaalsete maksimumide asukohad sarnasusmaatriksile, siis nagu näha Joonis 8, siis uudsusfunktsioonilt leitud lokaalsete maksimumide tipud märgivad üsna täpselt erinevate osade vahetumiskohad.



Joonis 8. Sarnasusmaatriks koos osade jaotustega.

Selle jaoks, et otsustada, millised leitud teemadest on omavahel sarnased loome uue maatriksi, kus anname igale sektsioonile väärtuse, milleks on antud juhul valitud sektsiooni väärtuste aritmeetiline keskmine, mis on seejärel normaliseeritud 0 ja 1 vahemikku. Joonis 9 on näha selle väärtustamise tulemus antud helifaili näitel, kus heledamad värvid tähistavad väiksemaid väärtusi ning tumedamad suuremaid.



Joonis 9. Heliteose osade väärtused maatriksil.

Viimase sammuna on vaja heliteose struktuuri saamiseks saadud osad klasterdada sarnasuse järgi, et leida korduvad teemad. Selle jaoks osutus parimaks lahenduseks SciPy teekide funktsioonid *scipy.cluster.hierarchy.linkage* [36] ning *scipy.cluster.hierarchy.fcluster* [37]. Esimese abil neist saadakse tulemuseks hierarhilise klasterduse tulemus aheldusmaatriksi kujul (*linkage matrix*). Funktsioon *fcluster* teisendab aheldusmaatriksi ühemõõtmeliseks massiiviks, kus igale heliteose osale on antud number vastavalt sellele, millisesse teemade klastrisse see kuulub. Muusikateoste vormianalüüsi korral on tavapärane tähistada erinevad teemad järjestikuste tähestiku tähtedega, seega konverteerime tulemuseks saadud arvumassiivi samale kujule. Brahmsi “Ungari tants nr. 5” korral on vormianalüüsi tulemusel saadud struktuuriks AABBCAABB.

Lõpptulemusena tagastatakse massiiv, kus on iga osa tähistatud vastava teema tähega ning iga teema kohta on antud ka algus ja lõpp. Brahmsi teost “Ungari tants nr 5” on kasutatud

vormianalüüsi näitena ka eelnevalt mainitud koodinäidete juures [38] ilmsestamaks töötluse oodatavat lõpptulemust, mis kattub antud algoritmi tulemusega.

3.4 Vormi ennustamine

Heliteose teemade ajalise liigenduse põhjal on võimalik teha järeldusi teose vormilise ehituse kohta. Klassikalises muusikas on vorme väga palju ning paljudel keerulisema ülesehitusega vormide puhul ei piisa alati ajalisest liigendusstruktuurist vormi kindlaks määratlemiseks. Selle tõttu on antud lõputöö raames valitud ennustamiseks tuntumad ja lihtsama ülesehitusega homofoonilised vormid, mis esinevad ka kõige tõenäolisemalt muusikahariduse õppekavades. Ennustatavad vormid ning nende iseloomustused on pärit Kerri Kotta muusikateooria õpikust [17]:

Kaheosaline lihtvorm –kahest erinevast osast vorm, tuntud ka kui AB-vorm. Vormi esimene pool põhineb mõnel klassikalise peateema vormil ning teine pool on üldiselt proportsionaalselt ligikaudu sama pikk teema, mis põhineb uuel materjalil või eelneva teema arendusel, mis ei ole tõlgendatav peateema tagasitulekuna. Kaheosalise lihtvormi puhul võib esineda ka teemade kordusi, näiteks võib vorm avalduda kujul AABB kus mõlemat osa korratakse.

Kolmeosaline lihtvorm – tuntud ka kui ABA-vorm. Tegu on klassikalise instrumentaalmuusika ühe tavapäraseima vormiga. Selle vormi esimeseks osaks on sarnaselt kaheosalisele lihtvormile peateema. Vormi keskosaks on kontrastne teema, kuhu üldreeglina tuuakse sisse uus muusikaline materjal ning vormi kolmas osa koosneb peateema tagasitulekust.

Rondo – vorm, mille puhul vaheldub peateema ehk refrään vaheosade ehk kupleedega. Selle vormi kõige sagedasem avaldumisvorm on viieosalise rondona (ABACA), mille puhul A tähistab peateemat ning B ja C vaheosasid. Rondo võib sageli avalduda ka seitsmeosalisena.

Rondo-sonaat – rondo vormi erikuju $ABACAB_1A$, mille puhul eelviimane kuplee B_1 tähistab uue muusikalise materjali asemel esimese kuplee juurde tagasipöördumist uues helistikus.

Variatsioonivorm – Selle vormi nimi tuleb ladinakeelsest sõnast *variatio* mis tähendab muutust või teisendust. Variatsioonivormi esimene osa on peateema eksposition, millele järgneb piiritlemata arv peateema variatsioone. Variatsioonide puhul jääb reeglina samaks teema vorm ning harmoonia, kuid muutub teema karakter, mida võib mõjutada nii peateemast erinev faktuur, dünaamika kui ka orkestratsioon jne.

Vormide ennustamisel on esmane eesmärk leida täpseid vasteid leitud vormistruktuurile antud vormide loendist, kuid see ei pruugi alati olla võimalik. Tihtipeale võivad vormid avalduda kergelt modifitseeritud kujudel ning samuti tuleb arvestada variandiga, et eelnevas vormianalüüsis võib leiduda ebatäpsusi. Samuti pole lõpprakenduses kasutajal mingeid piiranguid sellele, millist helifaili üles laadida ning selle tõttu ei pruugi heliteose vorm kuuluda üldse analüüsitavate hulka. Selle tõttu on antud rakenduse loomisel otsustatud täpse vaste puudumisel siiski ennustuse tulemusena tagastada lähima sobiva struktuuriga vorm.

Analüüsitava teose struktuur on eelneva vormianalüüsi tulemusena kujutatud tähtede jadana, kus iga täht tähistab erinevat teemat heliteoses. Analoogsel kujul saab kujutada ka analüüsiks valitud vorme, arvestades samas sellega, et ühele vormile võib vastata mitu erinevat struktuuri. Näiteks rondovormi võib avalduda nii 5-osalise (ABACA) kui ka 7-osalisena (ABACADA). Selle jaoks, et ennustada lähimat vormi saab kasutada teisenduskauguse ehk Levenshteini kauguse [39] leidmist. Vormi ennustamiseks kasutatav algoritm võrdleb erinevate vormide struktuure analüüsitava struktuuriga leides iga paari jaoks teisenduskauguse ning tagastades tulemuseks vähima teisenduskaugusega vormi kui parima pakkumise.

4. Veebirakenduse ülesehitus

Lõputöö raames loodud veebirakendus „Vormileidja“ on kättesaadav Heroku majutatud veebilehel[2] ning lähtekood on leitav GitHubi repositooriumist[3]. Käesolev peatükk annab ülevaate loodud veebirakenduse kasutajaliidesest ning funktsionaalsusest.

4.1 Disain ja arhitektuur

Veebirakendus on loodud, kasutades programmeerimiskeelt Python ning teeki Flask ning rakenduse eesliides (*front-end*) on loodud kasutades JavaScripti. Selleks, et tagada võimalus mitmel kasutajal rakendust korraga kasutada, on rakendus mitmelõimeline (*multithreaded*). Andmevahetus rakenduse tagaliidese ning eesliidese vahel toimub, kasutades JSON sõnumeid, mis sisaldab infot tempode, helistike ning vormi ülesehituse kohta.

Rakendus on loodud minimalistlikus disainis, kus põhirõhk oli teha kasutaja jaoks veebilehe kasutus võimalikult lihtsaks. Värvitoonidest on valitud mahedad värvid, mis on silmale kerge vaadata ning konkreetne värvipalett on loodud Coolors [40] lehe abil. Kasutatud fondid on valitud ning imporditud Google Fonts [41] rakenduse kaudu.

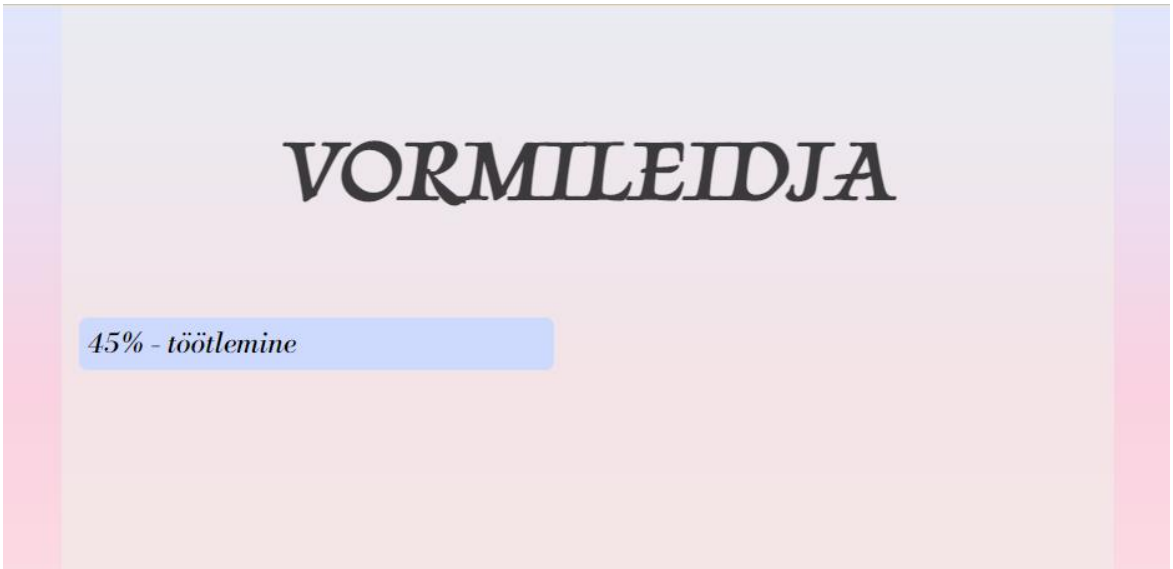
4.2 Avaleht

Veebilehe avalehel, mida on näha **Error! Reference source not found.**, on väike kirjeldus veebilehest. Lisaks on kasutajal on võimalus laadida oma arvutist üles vabalt valitud helifail, mis on .wav laiendiga. Seejärel laetakse fail üles ning alustatakse helifaili töötlemist. Töötlemise ajal on võimalik jälgida töötluse kulgu ekraanil oleva edenemisriba (*progress bar*) järgi, mida on ka näha Joonis 11. Heliteose analüüsimine, eriti suuremahuliste failide korral, võib võtta palju aega ning edenemisriba abil saab kasutaja lihtsamini jälgida veebirakenduse tööd ning parema aimduse, kui kaua hinnanguliselt veel aega võib minna.

[2] https://vormileidja.herokuapp.com/

[3] https://github.com/mannapuder/bakatoo

Joonis 10. Veebirakenduse avaleht.



Joonis 11. Veebirakendus faili töötlemise ajal.

4.3 Helifaili töötlemine

Helifaili töötlust jaguneb kolmeks suuremaks osaks: teose vormianalüüs ning vormi osade tempode ning helistike leidmine.

4.3.1 Vormianalüüs

Etteantud helifaili peal teostatakse vormianalüüs peatükis 3 kirjeldatud viisil. Esmalt leitakse heliteose struktuur ning selle põhjal ennustatakse ka kõige tõenäolisemalt heliteost kirjeldava vormi peatükis 3.4 kirjeldatud vormidest.

4.3.2 Tempo

MasterClass artikkel “Music 101: What Is Tempo? How Is Tempo Used in Music?” [42] defineerib tempo kui kiiruse, millel heliteost esitatakse. Tempot väljendatakse peamiselt kolmel erineval viisil: lööki minutis, itaaliakeelsed tempoterminid või loomuliku keeles iseloomustavad sõnad nagu näiteks aeglane või kiire. Kui lööki minutis annab väga täpse määratluse tempole, siis itaaliakeelsed terminid on subjektiivsemad ning defineerivad teatud kiiruste vahemikud.

Veebirakenduses leitakse heliteose iga osa tempo kasutades librosa teegi alammooduli *librosa.beat* funktsiooni *beat_track* [43], mis tagastab leitud tempo ühikutes bpm ehk *beats per minute.*

4.3.3 Helistik

Helistik on defineeritud Kerri Kotta muusikateooria õpikus [44] kui kindlalt helikõrguselt ehitatud duur- või mollhelilaad ning helistiku nimi koosneb vastava heli nimetusest ning laadi määratlusest. Näiteks helilt g ehitatud molli nimetus oleks g-moll jne.

Antud rakenduses leitakse teose osade helistikud kasutades Madmom teeki ning täpsemalt alammoodulit madmom.features.key [45]. Moodulis sisalduv funktsioon *CNNKeyRecognitionProcessor* ennustab teose või teose osa helistikku kasutades konvolutsioonilist närvivõrku järgides ideed Korzeniowski ja Widmeri teadusartiklist “Genre-Agnostic Key Classification With Convolutional Neural Networks” [46]. Närvivõrgu ennustuse tulemus on konverteeritud helistiku nimeks sama alammooduli funktsiooni *key_prediction_to_label* abil.

4.4 Kirjeldus loomulikus eesti keeles

Üheks eesmärgiks rakenduse puhul on töötlusel tuvastatud tunnused edastada märksõnade asemel kasutajale sidusa tekstina, mis annaks kasutajale lisainfot ning muudaks teose struktuuri mõistmise kergemaks. Kirjeldav tekst on enamjaolt genereeritud, kasutades keelemalle, kus eelnevalt koostatud lausetes on täidetud lüngad vastava infoga tempode, helistike ja vormi kohta.

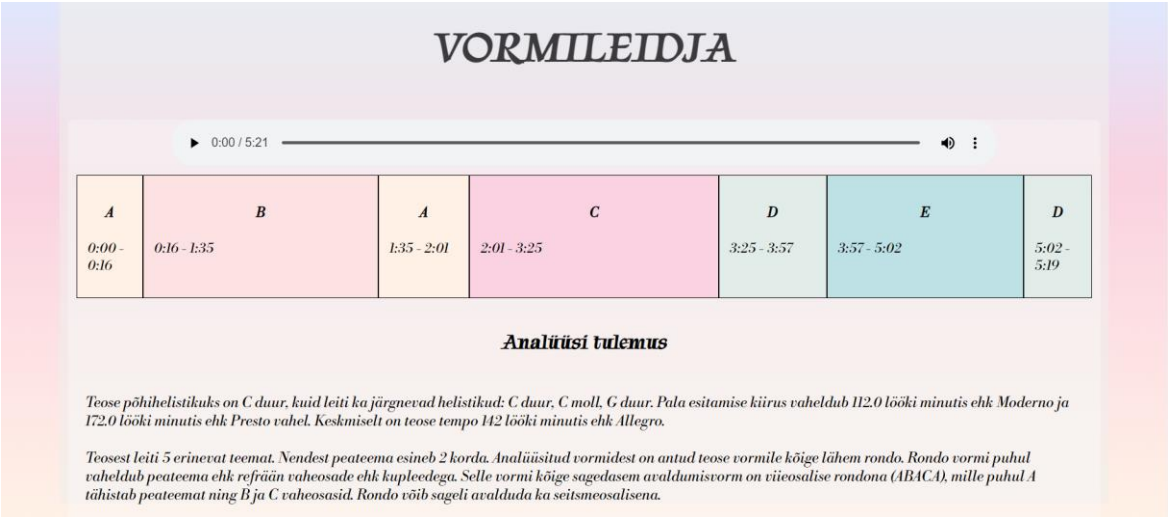
Leitud helistike põhjal valitakse sobivad laused teose kirjeldamiseks. Ühe helistiku korral täidetakse üksik lünk helistiku kohta, kuid kui teos sisaldab mitut helistikku, siis täidetakse lüngad vastavates lausetes nii teose põhihelistiku kui kõigi tuvastatud helistike kohta.

Tempo puhul analoogselt on kaks varianti sõltuvalt sellest kas tempo on teose jooksul ühtlane või vastasel juhul väljastatakse info nii tempode vahemiku kui teose keskmise tempo kohta. Tempode ühikuks on alati löökide arv minutis ning iga tempomärke puhul on juurde lisatud tempole vastav itaaliakeelne tempotermin [47], mis leitakse eelneva analüüsi põhjal leitud tempo järgi. Viimaseks genereeritakse laused ennustatud vormi kohta, mis sisaldavad lünki lisaks vormi nimetusele ka teemade arvu ning peateema korduste arvu kohta. Tulemusteksti lõppu lisatakse lühike kirjeldus ennustatud vormist. Vormide kirjeldused on kokku pandud Kerri Kotta muusikateooria õpiku põhjal [17].

4.5 Tulemuse esitus

Helifaili töötlemise järel kuvatakse kasutajale analüüsi põhjal saadud tulemus. Veebirakenduse tulemus on kujutatud Joonis 12, kus helifailina on ette antud Ludwig van Beethoveni teost „Rondo C-duuri op. 51 nr. 1“. Lehekülje ülemises osas on kasutajal võimalik helifaili mängida veebilehel kuvatud helipleieri kaudu. Kuvatud analüüsi tulemuse enda saab jagada kaheks põhiliseks osaks: visuaalne esitus vormist ning tekstiline iseloomustus.

Veebilehel kujutatakse teose vormi joonisena, kus iga erinev teema on eristatud erinevate värviga. Samuti on iga osa suurus graafikul proportsioonis vastava osa pikkusega kogu heliteoses ning iga osa juures on osa alguse ja lõpu ajatempel. Tänu sellele on kasutajal kergem mõista teose ülesehitust ning osade proportsioone ning soovi korral saab kasutaja ajatemplite järgi leida helipleieri kaudu kuulamiseks vastavad osad. Lisaks visuaalsele osale on juures teose omaduste kirjeldus.



Joonis 12. Beethoveni „Rondo C-duur op. 51 nr. 1“ analüüsi tulemus.

5. Tulemuse hindamine

Lõputöös valminud rakenduse hindamiseks sai rakendust antud testimiseks viiele erineva muusikalise taustaga kasutajale ning samuti testiti vormi ennustamise tulemust 10 klassikalise muusika teose näitel.

5.1 Tagasiside kasutajatelt

Veebirakendus anti testida viiele kasutajale eesmärgiga saada tagasisidet nii kasutajaliidese kui funktsionaalsuse osas. Kasutajate vanus oli vahemikus 15-28 ning muusikaline haridustase varieerus üldhariduskooli ning kõrghariduse vahel. Kasutajatel paluti hinnata rakenduse kasutamiskogemust, analüüsi tulemuste mõistetavust ning samuti tuua välja rakenduse puhul nii positiivsed küljed kui ka puudujäägid.

Kõik kasutajad olid ühel meelel, et rakendust on väga lihtne kasutada ning helifaili töötluse tulemus oli arusaadavalt esitletud. Kasutajad leidsid ka, et rakendus näeb kena välja ning teose struktuuri visuaalne välja toomine joonisena on kasulik ja annab hea ülevaate.

Rakenduse peamiseks puudujäägiks peeti seda, et rakendus on aeglane, kuna töötluseks kulub palju aega. Samuti toodi välja, et rakenduse vormianalüüsi täpsust saaks parandada, kuna mõnede heliteoste puhul ei leitud kõiki oodatud teemasid üles või olid ajamärked pisut ebatäpsed.

5.2 Vormi ennustamise testimine

Vormianalüüsi puhul on struktuuri korrektsuse testimine raskendatud mitmel põhjusel. Muusikateose vormianalüüsi puhul võivad ka muusikateooria haridusega inimesed jõuda sama teose puhul erinevatele järeldustele ning täpne teose struktuur võib mõnikord sõltuda konkreetsest salvestusest, juhul kui interpreet on teinud teose esitamisel enda jaoks sobivaid mugandusi. Samuti pole lõputöö raames suudetud leida avalikult ligipääsetavat andmestikku, mis sisaldaks vormi mõttes märgendatud heliteoste korpust. Sellise andmestiku puhul oleks oluline ka vastavate helifailide olemasolu, mille põhjal tulemused on saadud, eelnevalt mainitud põhjustel. Seega on kvantitatiivne testimine raskendatud, kuna see eeldaks eelnevalt vastava korpuse käsitsi märgendamist.

Alternatiivina on antud lõputöö raames teostatud kvalitatiivne testimine 10 teose põhjal. Testimiseks sai iga ennustatava vormi kohta valitud 2 vastavat vormi klassikalise muusika teost ning teose helifailide põhjal teostatud vormianalüüs ja ennustamise teose vormi, mida

seejärel võrreldi oodatud tulemusega. Korrektse vormi ennustamiseks peab ka eelnev struktuuri analüüsi tulemus olema piisavalt täpne ning seega saab vormi ennustamisel kaudselt testitud ka eelnev vormianalüüs, isegi kui puudub täpne oodatav tulemus struktuurile. Testimisel kasutatud audiofailid on leitavad lõputöö GitHub repositooriumi kasutast *test_audios*[4].

Tabel 1. Vormi ennustamise tulemused 10 teose näitel.

Testimiseks kasutatud heliteos	**Oodatav vorm**	**Ennustatud vorm**
Johann Sebastian Bach „Menuett G-Duur BWV Anh. 116“	Kaheosaline lihtvorm	Kaheosaline lihtvorm
Johann Sebastian Bach „Inglise süit Nr. 3 g-moll BWV 808 VI osa Gavott“	Kaheosaline lihtvorm	Kaheosaline lihtvorm
Frédéric Chopin „Prelüüd Des-Duur Op. 28 Nr. 15“	Kolmeosaline lihtvorm	Kolmeosaline lihtvorm
Joseph Haydn „Sümfoonia nr. 94 III osa“	Kolmeosaline lihtvorm	Kolmeosaline lihtvorm
Ludwig van Beethoven „Rondo C-duur Op. 51 nr. 1“	Rondo	Rondo
Wolfgang Amadeus Mozart „Sarvekontsert Nr. 3 KV.447 III osa Rondo“	Rondo	Rondo
Wolfgang Amadeus Mozart „Divertisment E-moll K. 563 VI osa Allegro“	Rondo-sonaat	Kolmeosaline lihtvorm

[4] https://github.com/mannapuder/bakatoo/tree/main/test_audios

Ludwig van Beethoven „Klaverisonaat Op. 13 Nr. 8 Pateetiline III osa Rondo-Allegro“	Rondo-sonaat	Rondo
Johann Sebastian Bach „Goldbergi variatsioonid“	Variatsioonivorm	Variatsioonivorm
Beethoven „Kuus lihtsat variatsiooni Šveitsi laulu teemale“	Variatsioonivorm	Rondo

Tabel 1 illustreerib testimisel saadud tulemusi. Kõige keerulisemateks vormideks tuvastada osutusid variatsioonivorm ning rondo-sonaat. Beethoveni teose „Kuus lihtsat variatsiooni Šveitsi teemale“ puhul tuvastas vormianalüüs piisava täpsusega variatsioonide algused ja lõpud helifailis, kuid probleem tekkis osade sarnasuste hindamisel. Variatsioonivormi puhul on tegu sama muusikalise materjaliga, mida esitatakse korduvalt pisut muudetud kujul, kuid antud teose töötlemise puhul tunnetati paljusid variatsioone liiga erinevana algsest teemast ning seetõttu hinnati neid peateemast erinevateks episoodideks, mis on just rondovormile omane struktuur, kus vahelduvad korduv refrään ning uue muusikalise materjaliga episoodid. Suure tõenäosusega mängis siin rolli variatsioonide suure varieeruvusega esituskiirused. Kuigi liigitamisel otsitava sarnasuse lävendit on võimalik algoritmis korrigeerida, siis mõjutaks selle lävendi tugev langetamine ka enamus ülejäänud testitud heliteoste tulemust, kuna nendel juhtudel ei pruugita enam erinevaid teemasid korrektselt eristada olukordades kus tõepoolest on tegu erineva muusikalise materjaliga.

Sarnane probleem tekib rondo ja rondo-sonaadi puhul, kuna need kaks vormi on omavahel väga sarnase struktuuriga ning ühe osa valesti liigitamine mõjutab juba lõpptulemust. Kuna rondo-sonaat on rondo erivorm, siis ei ole rondo pakkumine rondo-sonaadi asemel muusikateooria poole pealt vale vastus, kuigi ideaalis võiks siiski eristada, kummaga tegu. Mozarti „Divertisment E-moll K. 563 VI osa Allegro“ on ekslikult rondo-sonaadi asemel tuvastatud kui kolmeosaline lihtvorm, kuna oodatava rondo-sonaadi struktuuri ABACABA asemel tuvastati ABA. Antud juhul on tuvastas rakendus algse rondo-sonaadi struktuuri küll korrekt-

selt, kuid kuna struktuuri alguses ja lõpus kordub ABA, siis luges algoritm need üheks struktuuri osaks saadeski tulemuseks vastav ABA struktuur, kus A teema vastab rondo-sonaadi ABA struktuurile ning B teemana on tuvastatud rondo-sonaadi keskmine kuplee C.

Ülejäänud vormide puhul tuvastati antud näidete korral õiged vormid, kuigi mõnel juhul tuvastatud struktuur varieerus pisut vormi puhul reeglipärasest struktuurist, näiteks Chopini „Prelüüd Des-Duur Op. 28 Nr. 15“ puhul saadi tulemuseks oodatava ABA struktuuri asemel AAABA, kuna esimene peateema käsitlus sisaldab iseennast kordavat muusikalist materjali mis tuvastati kui uue teema algus. Selline erinevus ei takista aga enamjaolt vormi tuvastamist kuna selle lõputöö raames käsitletavate vormide puhul loeb vormi tuvastamisel eelkõige erinevate teemade esinemise järjestus mitte tingimata kui mitu korda sama teemat korratakse. Klassikalise muusika teoste puhul esineb lisaks reeglipärase ehitusega teostele ka palju erandeid, millega pole alati võimalik automaatse vormianalüüsi puhul arvestada ning seetõttu on eeldatav, et rakendus võib aeg-ajalt eksida. Testimise tulemuste põhjal saab aga väita, et antud näidete puhul on algoritm võimeline enamus juhtudel teose vormi korrektselt ennustama ning seda ka juhul kui teose struktuuris on väiksemaid ebatäpsusi.

6. Edasiarendamise võimalused

Üks peamiseid arenduskohti veebirakenduse jaoks oleks kindlasti töötluseks kuluva aja optimeerimine. Hetkel on suuremahuliste helifailide töötlemine väga aeganõudev, kuigi on kasutatud mitut lõime. Samuti saaks kasutaja jaoks veebilehe kasutamist muuta mugavamaks, kui toetada rohkemate laienditega helifailide töötlust. Rakenduses „Vormileidja“ tugineb loomulikus keeles tulemus piiratud valikule keelemallidele ning seetõttu on teksti ülesehitus enamjaolt iga kord sama.

Hetkel kasutusel olev vormianalüüsi algoritm suudab tuvastada vaid valitud hulka homofoonilistest vormidest, seega oleks tulevikus võimalik lisada juurde rohkemate vormide äratundmist ning lisaks sellele saaks parandada ka olemasoleva algoritmi täpsust. Vormide lisamisel peaks kaaluma nende eristamiseks peale teemadel põhineva struktuuri ka teiste tunnuste analüüsimist, kuna keerulisemate vormide puhul ei pruugi olla võimalik vaid struktuuri põhjal vorme piisava kindlusega eristada. Selleks peaks lisama uute tunnuste tuvastamise ja analüüsimise. Ühe variandina saaks analüüsida heliteoses kasutatud akorde ning nende põhjal tuvastada ja analüüsida heliteoses leiduvaid harmoonilisi järgnevusi, mis on teatud vormidele või vormi osadele iseloomulikud.

Samuti on antud lõputöö raames kasutatud vormianalüüsi puhul tehtud eeldus, et korraga kõlab maksimaalselt üks teema, mille tõttu on antud juhul ongi võimalik vaid homofooniliste vormide ära tundmine. Tulevikus oleks võimalik funktsionaalsust laiendada, tuues sisse ka polüfooniliste teemade ära tundmist, kasutades maatriksfaktoriseerimist (ingl *matrix factorization*) meloodiate eraldamiseks [48].

Kokkuvõte

Muusikateoste käsitsi analüüsimine on aeganõudev protsess, mis tihti eeldab ka paljusid eelteadmisi, eriti klassikalise muusika vormianalüüsi puhul. Tihti puudub ka võimalus iseseisva analüüsi tulemusi kontrollida, kuna puuduvad seda toetavad materjalid.

Antud lõputöös anti ülevaate olemasolevatest muusikateose kirjeldamise rakendustest ning nende tugevustest ja puudujääkidest. Samuti analüüsime olemasolevaid uurimustöid heliteose vormi leidmiseks.

Bakalaureusetöö eesmärk luua algoritm heliteose muusikalise vormi ennustamiseks ning luua loodud lahendust kasutav veebirakendus sai täidetud. Loodud rakendus „Vormileidja“ on teadaolevalt esimene eesti keelne veebirakendus, mille puhul on võimalik üleslaadida oma helifaili, et saada selle vormilist jaotust ja ennustust vormi kohta. Rakendust testinud kasutajad leidsid, et rakendus on kasutajasõbralik ning tulemuse esitus aitab kaasa heliteose struktuuri mõistmisele.

Muusika automaatne vormianalüüs on ala, kus on veel palju arenguruumi. Tulevikus saaks rakendust edasi arendada parandades tulemuse täpsust ning lisades võimalusi suurema hulga muusikavormide tuvastamiseks.

Kasutatud allikad

[1] „GetSongKey,“ https://getsongkey.com/. (10.01.2022)

[2] „GetSongbpm,“ https://getsongbpm.com/. (10.01.2022)

[3] „Chosic,“ https://www.chosic.com/music-genre-finder/. (11.01.2022)

[4] „ChordAI,“ https://www.chordai.net/. (09.01.2022)

[5] M. P. Murumaa, „Muusikateose automaatne kirjeldamine loomulikus eesti keeles,“ Tartu, 2022.

[6] V. Tõnisson, Automatic Description of Music in Natural Estonian, Tartu, 2021.

[7] M. S. Roberts, „Classical music research shows huge surge in young people streaming Mozart - Classic FM,“ Classic FM, 19.08 2020. https://www.classicfm.com/music-news/surge-millennial-gen-z-streaming-classical-music/. (20.04.2022)

[8] „librosa,“ librosa development team, https://librosa.org/doc/latest/index.html. (11 01.2022)

[9] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg ja O. Nieto, „librosa: Audio and Music Signal Analysis in Python,“ %1 *14th Python in Science Conference (SciPy 2015)*, Texas, 2015.

[10] „Madmom documentation,“ https://madmom.readthedocs.io/en/latest/. (15.04.2022)

[11] „SciPy,“ https://scipy.org/. (05.05.2022)

[12] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. v. d. Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov ja M. Nikolay, „SciPy 1.0: fundamental algorithms for scientific computing in Python,“ *Nature Methods*, kd. 17, pp. 261-272, 2020.

[13] „Design Decisions in Flask - Flask Documentation (2.1.x),“ https://flask.palletsprojects.com/en/2.1.x/design/. (25.03.2022)

[14] V. Singh, „Flask vs Django in 2022: Which Framework to Choose?,“ 07.01.2022 https://hackr.io/blog/flask-vs-django. (25.03.2022)

[15] „Heroku,“ https://www.heroku.com/. (09.01.2022)

[16] „Home - Docker,“ https://www.docker.com/. (08 05.2022)

[17] K. Kotta, „VI.1 Põhimõisted - Muusikateooria õpik,“ http://mt.ema.edu.ee/vi-muusika-vormilised-struktuurid/vi-1-pohimoisted/. (20.04.2022)

[18] V. Jayaram, „Pychorus: Python module for detecting musical choruses,“ https://github.com/vivjay30/pychorus. (10.03.2022)

[19] M. Goto, „A chorus section detection method for musical audio signals and its application to a music listening station,“ *IEEE Transactions on Audio, Speech and Language Processing,* kd. 14, nr 5, pp. 1783-1794, 2006.

[20] V. Jayaram, „Finding Choruses in Songs with Python,“ https://towardsdatascience.com/finding-choruses-in-songs-with-python-a925165f94a8. (10.03.2022)

[21] l. d. team, „Laplacian segmentation - librosa-gallery 0.1.0 documentation,“ https://librosa.org/librosa_gallery/auto_examples/plot_segmentation.html#sphx-glr-auto-examples-plot-segmentation-py. (05.03.2022)

[22] B. McFee ja D. P. Ellis, „Analyzing song structure with spectral clustering,“ *ISMIR,* Taipei, 2014.

[23] M. Müller, Fundamentals of Music Processing -- Using Python and Jupyter Notebooks, Springer Verlag, 2021.

[24] „C4 - Music Structure Analysis,“ https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4.html. (05.05.2022)

[25] J. Serrà, M. Müller, P. Grosche ja J. L. Arcos, „Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity,“ *IEEE Transactions on Multimedia,* kd. 16, nr 5, pp. 1229-1240, 2014.

[26] librosa development team, „librosa.effects.trim - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.effects.trim.html. (07.05.2022)

[27] librosa development team, „librosa.effects.harmonic - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.effects.harmonic.html. (08.05.2022)

[28] librosa development team, „librosa.feature.chroma_cqt - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.feature.chroma_cqt.html. (07.05.2022)

[29] librosa development team, „librosa.decompose.nn_filter - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.decompose.nn_filter.html. (08.05.2022)

[30] librosa development team, „librosa.segment.recurrence_matrix - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.segment.recurrence_matrix.html. (07.05.2022)

[31] librosa development team, „librosa.segment.path_enhance - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.segment.path_enhance.html. (07.05.2022)

[32] librosa development team, „librosa.segment.recurrence_to_lag - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.segment.recurrence_to_lag.html. (08.05.2022)

[33] The SciPy Community, „scipy.ndimage.gaussian_filter1d - SciPy v1.8.0 Manual,“ https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter1d.html. (09.05.2022)

[34] „C4S4_StructureFeature,“ https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S4_StructureFeature.html. (03.05.2022)

[35] The SciPy community, „scipy.ndimage.median_filter - SciPy v1.8.0 Manual,“ https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.median_filter.html. (08.05.2022)

[36] The SciPy community, „scipy.cluster.hierarchy.linkage - SciPy v1.8.0 manual,“ https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html. (07.05.2022)

[37] The SciPy community, „scipy.cluster.hierarchy.fcluster - SciPy v1.8.0 Manual,“ https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html. (08 05.2022)

[38] „C4S4_Novelty segmentation,“ https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S4_NoveltySegmentation.html. (07.05.2022)

[39] „Levenshtein distance,“ https://en.wikipedia.org/wiki/Levenshtein_distance. (08 05.2022)

[40] „Coolors - The super fast color palettes generator!,“ https://coolors.co/. (25.04.2022)

[41] „Google Fonts,“ https://fonts.google.com/. (06 05.2022)

[42] „Music 101: What Is Tempo? How Is Tempo Used in Music?,“ 05.08 2021. https://www.masterclass.com/articles/music-101-what-is-tempo-how-is-tempo-used-in-music. (01 05.2022)

[43] „librosa.beat.beat_track - librosa 0.9.1 documentation,“ https://librosa.org/doc/main/generated/librosa.beat.beat_track.html. (29.04.2022)

[44] K. Kotta, „I.5.Laad - Muusikateooria õpik,“ http://mt.ema.edu.ee/muusika-elementaarteooria/i-5-laad/. (29.04.2022)

[45] „madmom.features.key - madmom 0.16 documentation,“ https://madmom.readthedocs.io/en/v0.16/modules/features/key.html. (23.04.2022)

[46] F. Korzeniowski ja G. Widmer, „Genre-Agnostic Key Classification With Convolutional Neural Networks,“ %1 *19th International Society for Music Information Retrieval Conference*, Paris, 2018.

[47] „Tempo - Vikipeedia,“ https://et.wikipedia.org/wiki/Tempo. (09.05.2022)

[48] T. Nakamura ja H. Kameoka, „Shifted and convolutive source-filter non-negative matrix factorization for monaural audio source separation,“ %1 *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[49] M. Ardi, „Musical Instrument Sound Classification using CNN,“ 19.08.2020. https://becominghuman.ai/musical-instrument-sound-classification-using-cnn-part-1-2-43197e554cc8. (28.04.2022)

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Anna Talas,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Muusikateose vormianalüüs loomulikus eesti keeles**, mille juhendaja on Sven Aller, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Anna Talas

10.05.2022