https://doi.org/10.7840/kics.2023.48.11.1500

음원 분리 및 자동 채보 학습을 활용한 음악 유사성 분석 시스템 구현

구 연 우◆, 이 재 호°

Implementation of Music Similarity Analysis System Employing Source Separation and Automatic Transcription Learning

Yeonwoo Gu◆, Jaeho Lee°

요 약

기존의 음악 유사성 분석 방식은 보편적으로 모든 악기와 오디오 채널이 포함된 혼합 음악 데이터를 기반으로 진행되는 특성이 있기에, 템포가 빠른 댄스 장르의 음악을 느리게 편곡하여 발라드 장르로 변경하는 등 원곡에서의 장르 혹은 가창자 등에 변화가 발생할 경우 자동 표절 판별이 난해하다. 이러한 기존 유사성 분석 방식의 개선을 위해 본 논문에서는 기존 연구에서 제안한 음악 유사성 분석 시스템의 전체 구조를 기반으로 적합한 오픈 소스를 탐색하고, 이를 활용해 실제 분석 시스템을 구현하였다. 오픈 소스 딥러닝 모델의 활용을 위해 음악 데이터셋을 구축하고 데이터를 전처리하여 학습을 진행한 결과 음원 분리 모델인 Spleeter에서 SDR 5.439를, 음악 자동 채보 모델인 Omnizart vocal에서 F1-score 0.853을 달성하였다.

키워드 : 음악 유사성, 음원 분리, 자동 채보, 인공 신경망
Key words : Music Similarity, Source Separation, Automatic Music Transcription, Artificial Neural Networks

ABSTRACT

Since existing music similarity analysis method are typically based on mixed music data that includes all instruments and audio channels, it is difficult to automatically detect plagiarism when there is a change in the genre or singer from the original song, such as slowing down a fast-tempo dance song and changing it to a ballad genre. In order to improve these existing similarity analysis method, this paper explored suitable open sources based on the overall structure of the music similarity analysis system proposed in previous studies, and uses them to implement an actual analysis system. To utilize open source deep learning models, we built a music dataset and preprocessed the data for training, achieving SDR of 5.439 for spleeter, a music separation model, and F1-score of 0.853 for Omnizart vocal, a music transcription model.

Ⅰ. 서 론

음악 표절은 창작자의 지적 재산권을 침해하는 행위로, 창작자의 노력과 창의성을 무시하는 것뿐만이 아닌 창작자에게 경제적으로 손해를 입힐 수 있다. 음악 표절은 저작권법에 따라 처벌될 수 있으며, 음악 산업의 발

※ 본 연구는 2023년도 덕성여자대학교 교내연구비 지원으로 이루어졌음.
◆ First Author : Duksung Women's University, Department of Software, yw0512@duksung.ac.kr, 학생회원
° Corresponding Author : Duksung Women's University, Department of Software, izeho@duksung.ac.kr, 종신회원
논문번호 : 202308-025-C-RN, Received August 1, 2023; Revised August 30, 2023; Accepted August 31, 2023

전을 저해하는 요인으로 작용하기도 한다. 음악 표절 분석에 있어 사람의 주관적 판단에 의지하는 것은 객관성 확보가 어렵다. 반면 알고리즘을 기반으로 하는 유사성 분석 방식은 주로 멜로디를 기준으로 하여 유사성을 판단하기에, 멜로디 외에도 화성, 리듬 등 음악의 다양한 특성을 정확하게 파악하고 분석하는데 있어 어려움을 겪을 수 있다. 또한 대부분 음원 분리를 수행하지 않고 모든 악기와 오디오 채널이 포함되어 있는 혼합 음악 데이터를 기반으로 유사도 분석을 진행하기에, 원곡의 템포, 가창자 등을 변화시켜 타 장르로의 편곡이 발생할 경우 이를 판별하기 어렵다는 단점 또한 존재한다. 이러한 기존의 분석 방식의 단점을 개선하기 위해, 본 논문에서는 더 정확한 분석을 위해 혼합 음악 데이터에서 멜로디, 리듬, 화성의 음원 분리를 진행하고, 분리한 음원 데이터의 정량화를 위해 채보 기술을 도입하여 인공지능 기반의 유사성 분석 시스템을 구현하고자 하였다.

따라서 본 논문에서는 기존 논문[1]에서 제안한 음악 유사성 분석 시스템의 구조를 기반으로 음원 분리 및 자동 채보 과정을 인공지능 모델을 통해 수행하고, 멜로디와 리듬 및 코드 패턴에 따른 음악 유사성을 체계적으로 분석하고자 하였다. 또한 편곡에 대한 유사성 분석의 강인성을 확보하기 위해 모든 학습 데이터에 대해 다수의 커버곡을 생성하여 학습에 활용하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 음원 분리, 자동 채보에서 자주 활용되는 Open-Unmix와 MT3 및 유사성 분석 단계에서 활용 가능한 ChromaCoverID에 대해 설명한다. Ⅲ장에서는 본 논문에서 활용한 오픈소스 모델 및 알고리즘에 대해 설명하고, 시스템의 전체적인 구조와 진행 과정에 대해 서술한다. Ⅳ장에서는 선정한 모델의 학습을 위해 구축한 데이터셋과 모델 학습 및 학습 결과를 보이고, 실제 구현된 시스템을 통해 얻은 유사성 결과를 분석한다. 마지막으로 Ⅴ장에서는 본 논문에서 구현한 음악 유사성 시스템에 대해 요약하고, 결론 및 추후 계획에 대해 기술하고자 한다.

Ⅱ. 연구 배경

2.1 Open-Unmix

Open-Unmix[2]는 딥러닝을 기반으로 음악 데이터에서 보컬 및 각종 악기를 분리하는 다중 소스 분리를 목표로 하는 프로젝트이며, MUSDB18 공개 데이터셋을 통한 사전 학습을 진행한 모델을 제공한다. 해당 모델을 통해 여러 가지 트랙이 혼합된 음악 데이터에서 보컬, 드럼, 베이스, 그 외 악기 총 4가지의 트랙으로

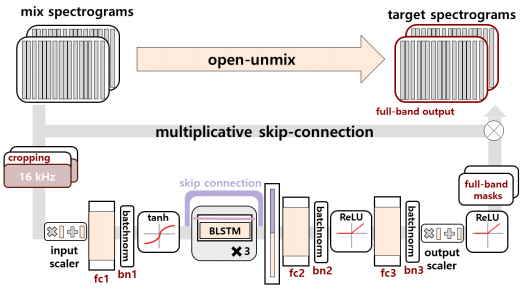

그림 1. Open-Unmix의 모델 구조[3]
Fig. 1. The Model Architecture of Open-Unmix

분리가 가능하다. Open-Unmix의 모델 구조는 그림 1과 같으며, Short-Time Fourier Transform(STFT)을 통해 변환된 Spectogram을 바탕으로 3계층 양방향 LSTM 네트워크를 사용해 학습을 수행한다.

2.2 MT3

MT3[4]는 Transformer 기반의 T5 아키텍처를 기반으로 하는 사용하는 다중 악기 자동 채보 모델이며, MusicNet, MAESTROv3, Slakh2100, GuitarSet 등의 다양한 데이터셋을 활용하여 학습을 수행하였다. T5 아키텍처는 Encoder-Decoder Transformer 모델로, 2017년 Vaswani et al.의 모델을 사용했다. MT3의 Transcription 과정은 그림 2와 같다. MT3는 내부적으로 Source Seperation 기능을 탑재하고 있으며, 이를 기반으로 자동 음악 채보 기능을 수행한다.

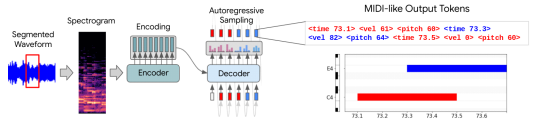

그림 2. MT3의 Transcription 과정[5]
Fig. 2. Transcription Process of MT3

2.3 ChromaCoverID

ChromaCoverID[6]는 커버곡 식별 작업을 위한 다양한 Chroma 및 오디오 유사도 측정값을 계산하는 함수 및 방법을 모아둔 Github 오픈소스이다. ChromaCoverID에는 QMax[7], DMax[8] 커버곡 유사도 측정 방법이 파이썬으로 구현되어 있다. QMax는 실제 커버 송을 올바르게 식별한 비율을 나타내며, QMax의 값이 높을수록 커버 송을 높은 정확도로 식별하고 있음을 의미한다. DMax는 커버 송 식별 성능의 또 다른 평가 지표로, 실제 원곡을 올바르게 식별한 비율을 나타낸다. DMax의 값이 높을수록 실제 원곡을 높은 정확도

로 식별하고 있음을 의미한다.

음악의 유사도를 분석하는 방식에는 방금 언급한 ChromaCoverID를 제외하고도 다양한 연구[9-10]들이 존재하나, 앞서 서론에서 언급했듯 음원 분리를 수행하지 않은 혼합 음악 데이터를 기반으로 진행되거나 멜로디를 기준으로 하여 유사도를 분석하기에 음악이 가진 다양한 특성을 반영한 분석과 기존 곡의 편곡을 탐지하기 어렵다는 등의 문제점을 지닌다. 이와 같은 단점을 보완하기 위해서는 혼합 음악 데이터를 멜로디, 리듬, 화성 등 여러 트랙으로 분리하는 음원 분리가 선행되어야 한다. 따라서 본 연구에서는 음원 분리와 자동 채보 생성을 결합한 음악 유사성 분석을 수행함으로써 음악의 다양한 특성을 고려하며 더 정확하고 포괄적인 유사도 분석을 제공하고자 하였다. 이외에도 머신러닝 및 딥러닝을 활용하여 다양한 음악이 가진 특성을 자동으로 학습하고 이를 판별하는 방식 또한 고려해볼 수 있다.

Ⅲ. 분석 시스템 설계

3.1 활용 기술

본 논문은 이번 장에서 언급하는 Spleeter, Omnizart, MelodyShape, GrooveToolBox 총 4개의 오픈 소스 모델 및 알고리즘을 각 단계에서 활용하여 음악 유사성 분석 시스템을 설계 및 구현하였다.

3.1.1 Spleeter

Spleeter[11]는 딥러닝 기반의 음원 분리를 위한 오픈 소스 파이썬 라이브러리로, 사전 학습된 모델을 이용해 모든 트랙이 혼합되어있는 음악 파일에서 보컬, 드럼, 베이스, 피아노 등의 개별 트랙(stems)을 분리하여 추출해 내는 것을 목표로 한다. Spleeter의 딥러닝 모델 구조는 U-Net 아키텍처를 기반으로 하며, 해당 구조에서 인코더 층과 디코더 층이 각각 6개씩 구성되어 총 12개의 층으로 구성되어 있다. 사전 학습 모델은 손실 함수로 마스크된 입력 혼합 Spectrogram과 소스-타겟 Spectrogram 사이의 L1 Loss를, 활성화 함수로는 Adam을 사용하여 모델을 학습하였고, 소프트 마스킹 또는 다중 채널 Wiener 필터링을 사용하여 학습된 모델에서 추정한 소스 스펙트로그램에서 분리를 수행한다.

3.1.2 Omnizart

Omnizart[12]는 Automatic Music Transcription (AMT)을 위한 파이썬 라이브러리로, 보컬. 드럼, 코드, 피아노 솔로 등의 다양한 음악 콘텐츠에 대한 Tensorflow 기반의 자동 채보 딥러닝 모델을 제공한다. 보컬 채보 모델에는 피치 추출기와 음표 분할 모듈이 존재한다. 피치 추출기는 사전 훈련된 Patch-CNN이 활용되었고, 음표 분할 모듈은 반지도 학습이 가능한 가상 적대적 훈련을 사용한 PyramidNet-110 및 ShakeDrop 정규화를 기반으로 구현되었다.

드럼 채보 모델은 2021년 Wei et al.의 모델을 재구현한 것으로, 컨볼루션 레이어와 Attention 매커니즘이 네트워크에 포함된다. 코드 채보 모델은 Harmony Transformer(HT)를 사용하여 구현되었다. HT 모델은 인코더-디코더 아키텍처를 기반으로 하며, 인코더는 입력에 대한 코드 분할을 수행하고 디코더는 해당 분할 결과를 기반으로 코드 진행을 인식한다.

3.1.3 음악 유사성 판단 알고리즘

MelodyShape[13]은 음악 작품 간의 멜로디 유사성을 계산하기 위한 오픈 소스 자바 라이브러리로, 멜로디의 음높이-시간 평면에서 나타나는 기하학적 모양을 기반으로 설명되는 유사성을 계산하는 여러 알고리즘을 구현하였다. GrooveToolBox[14]는 Symbolic 드럼 루프의 분석과 비교, 리듬의 특성, 마이크로 타이밍 특성, 유사성 지표를 계산하는 파이썬 라이브러리이며, Groove에 대한 해밍 거리(Hamming Distance), 퍼지 해밍 거리(Fuzzy Hamming Distance) 및 구조적 리듬 유사성의 3가지 유사성 메트릭을 포함하고 있다.

3.2 시스템 구조 및 설계

본 논문에서 제안하는 음악 유사성 분석 시스템의 전체 구조도는 그림 3과 같다. Pulse Code Modulation(PCM) 형태의 혼합 음악 데이터를 입력 데이터로 받아 음원 분리, 자동 채보, 유사도 측정의 3단계를 거쳐 각 트랙별 유사도를 측정한다. 이때 위의 활용 기술에서 소개한 Spleeter와 Omnizart 모델을 각각 음원 분리와 자동 채보 단계에서, MelodyShape과 GrooveToolBox를 유사성 판별 단계에서 사용하였다.

서론에서 언급한 바와 같이, 기존 음악 유사성 분석은 같은 곡에서 가창자, 템포 등의 변동 사항이 생길 경우 그 둘의 유사성을 인지하지 못한다는 문제점이 있다. 따라서 정확한 유사성 분석을 위해서는 혼합 음악 데이터에서 각 트랙별 음원 분리가 선행되어야 한다.

본 논문에서는 보컬과 각종 악기가 혼합되어 있는 음원 데이터를 사전에 탐색한 Spleeter 모델을 사용하여 보컬, 드럼, 코드 세 개의 트랙으로 분리하는 과정을 수행하도록 설계하였다. Spleeter 모델은 2, 4, 5 stems로 분리하도록 설정이 가능한데, 원하는 트랙별 분리를 위해 4stems 모델을 선정하였다. 각 stems 별 분리할

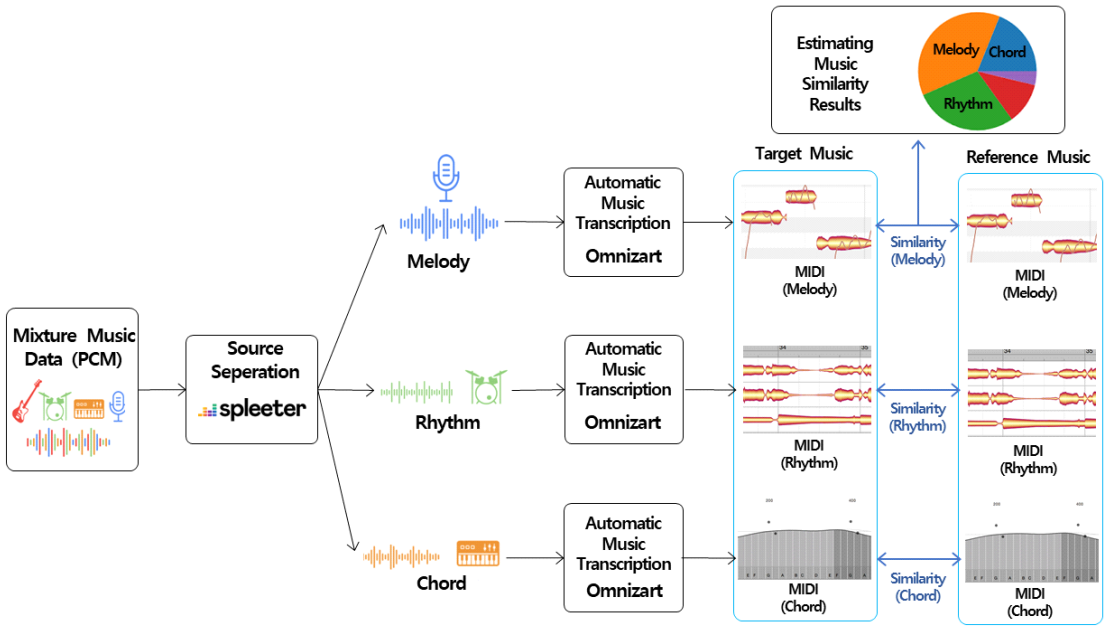

그림 3. 음악 유사성 분석 시스템의 전체 구조도
Fig. 3. The overall structure of the music similarity analysis system

수 있는 트랙의 종류는 표 1과 같다.

표 1. Spleeter의 stems 별 분리 음원
Table 1. Spleeter's stems-specific separation music

stems	output
2	Vocal, Accompaniment
4	Vocal, Drum, Bass, Other
5	Vocal, Drum, Bass, Piano, Other

기존의 음원 데이터가 음원 분리 과정을 거치면 Vocal, Drum, Bass, Others의 4개 영역으로 분리된 음원이 생성되며, 이 4개 영역 중 others를 제외한 3개의 영역만을 사용한다. 유사성 판단을 위해서는 채보를 통해 전자 악보를 생성하는 것이 선행되어야 하기에, 해당 분리 음원을 Omnizart 모델에 입력 데이터로 넣어주면 AMT를 통해 Musical Instrument Digital Interface (MIDI) 형식의 전자 악보를 생성한다.

위의 두 과정을 거친 원본 데이터는 세 영역으로 분리되고, 최종적으로 MIDI 형식의 전자 악보가 된다. 변환된 전자 악보를 기반으로 레퍼런스 음원의 각 악보 데이터와 비교하여 유사성 검토를 진행한다. 멜로디와 화음 트랙은 MelodyShape을 이용하였으며, 리듬에 해당하는 드럼 트랙은 GrooveToolBox를 사용하여 유사도 측정을 진행하였다.

Ⅳ. 구현 및 성능 분석

4.1 음원 데이터 구축

Spleeter와 Omnizart 모델의 학습 및 평가를 위해 음원 데이터를 수집하여 데이터셋을 구축하였다. 두 모델의 학습 및 테스트 과정에서 원곡을 사용할 경우 저작권과 관련된 문제가 발생할 수 있으므로, 가창자를 변경하여 원곡과 최대한 유사한 형태로 수집하였다. 발라드, 트로트, 댄스, Rock, 힙합, RnB의 총 6가지 장르의 곡을 다양하게 수집하였으며, 장르 별로 수집한 원곡과 커버곡 데이터의 개수는 표 2와 같다. 또한 서론에서 언급한 바와 같이, 편곡에 대한 유사성 분석의 강인성을 확보하기 위해 하나의 곡을 대상으로 다양한 장르로 변경시킨

표 2. 장르 별 수집한 데이터의 개수
Table 2. Number of data collected for each genre

Genre	Original	Cover	Org+Cover
Ballade	300	2,700	3,000
Dance	200	1,800	2,000
Trot	200	1,800	2,000
HipHop	100	900	1,000
Rock	100	900	1,000
RnB	100	900	1,000
Total			10,000

편곡 데이터를 생성하여 곡 당 총 10개의 데이터를 가지도록 하였다. 또한 Spleeter와 Omnizart 모델의 Train, Validation, Test 데이터의 개수는 각각 6,300개, 2,700개, 1,000개로 설정하였다.

4.2 데이터 전처리

4.1에서 구축한 음원 데이터를 활용하여 Spleeter와 Omnizart 모델의 학습과 성능 평가를 위해 각 모델 별 학습 데이터 형식에 맞게 데이터 전처리를 진행하였다. Spleeter, Omnizart 모델의 학습을 위한 환경 정보는 표 3과 같다.

Spleeter 4stem 모델의 학습을 위해서는 각 곡의 wav 파일과 config.json 파일 및 2개의 csv 파일이 필요하다. wav 파일의 경우, 각 곡마다 mixture, vocal, drum, bass, other 총 5개의 wav 파일이 필요하다. 이때 모든 파일의 길이가 동일해야 하며, Mono가 아닌 Stereo 타입이어야 학습 시 사용이 가능하다.

config.json 파일은 learning rate, batch size 등의 모델 파라미터 정보, checkpoint step, summary step, 두 개의 csv 파일이 저장되어 있는 경로 등 각종 정보들로 구성되어 있다. 해당 파일에서 csv 파일 저장 위치를 학습 환경에 맞게 수정해야 하며, 그 외에도 checkpoint step 등을 원하는 대로 수정할 수 있다. config.json의 구성 정보는 그림 4와 같다.

train_csv, validation_csv 각각의 파일에는 Spleeter 모델 학습에 사용되는 5개의 wav 파일이 위치한 경로와 해당 곡의 duration 정보로 구성되어 있다. 각 csv 파일의 구성 정보는 그림 5와 같다.

Omnizart는 Spleeter와 달리 트랙 별로 모델 구조가 상이하며, 본 논문에서는 Omnizart Vocal, Drum, Chord 모델을 사용했다. Omnizart 모델의 학습 및 테스트를 위해서는 feature 파일을 생성해야 하는데, feature는 인공지능 모델 입력에 사용하기 위해 음악 데이터를 특징 벡터로 변환한 것이며, Hierarchical Data Format(HDF) 확장자를 가진다.

보컬 채보 모델의 경우 원본 wav 파일에서 보컬 트랙을 추출한 뒤, 추출한 보컬 분리 음원의 Pitch 값을

표 3. 모델 학습 환경
Table 3. Model Training Environment

S/W	Model	version
python	Spleeter	3.7.13
	Omnizart Vocal	3.6.10
	Omnizart Drum	3.7.0
	Omnizart Chord	3.6.13
tensorflow	Spleeter	2.10.0
	Omnizart Vocal	2.3.0
	Omnizart Drum	2.5.0
	Omnizart Chord	2.5.0
CUDA	ALL	11.3.1
CuDNN	ALL	8.2.1


```
{
  "train_csv": "./configs/musdb_train.csv",
  "validation_csv":
"./configs/musdb_validation.csv",
  "model_dir": "model",
  "mix_name": "mix",
  "instrument_list": [ "vocals", "drums", "bass",
"other" ],
  "sample_rate": 44100,
  "frame_length": 4096,
  "frame_step": 1024,
  "T": 512,
  "F": 1024,
  "n_channels": 2,
  "separation_exponent": 2,
  "mask_extension": "zeros",
  "learning_rate": 1e-4,
  "batch_size": 4,
  "training_cache": "./cache/training",
  "validation_cache": "./cache/validation",
  "train_max_steps": 200000,
  "throttle_secs": 1800,
  "random_seed": 3,
  "save_checkpoints_steps": 1000,
  "save_summary_steps": 5,
  ...
}
```


그림 4. config.json의 구성 정보
Fig. 4. Configuration information in the config.json

mix_path	vocals_path	drums_path	bass_path	other_path	duration
train/1/mixture.wav	train/1/vocals.wav	train/1/drums.wav	train/1/bass.wav	train/1/other.wav	267
train/2/mixture.wav	train/2/vocals.wav	train/2/drums.wav	train/2/bass.wav	train/2/other.wav	182
train/3/mixture.wav	train/3/vocals.wav	train/3/drums.wav	train/3/bass.wav	train/3/other.wav	196
train/4/mixture.wav	train/4/vocals.wav	train/4/drums.wav	train/4/bass.wav	train/4/other.wav	216
train/5/mixture.wav	train/5/vocals.wav	train/5/drums.wav	train/5/bass.wav	train/5/other.wav	193
train/6/mixture.wav	train/6/vocals.wav	train/6/drums.wav	train/6/bass.wav	train/6/other.wav	218
train/7/mixture.wav	train/7/vocals.wav	train/7/drums.wav	train/7/bass.wav	train/7/other.wav	218
train/8/mixture.wav	train/8/vocals.wav	train/8/drums.wav	train/8/bass.wav	train/8/other.wav	194
train/9/mixture.wav	train/9/vocals.wav	train/9/drums.wav	train/9/bass.wav	train/9/other.wav	221
train/10/mixture.wav	train/10/vocals.wav	train/10/drums.wav	train/10/bass.wav	train/10/other.wav	238
train/11/mixture.wav	train/11/vocals.wav	train/11/drums.wav	train/11/bass.wav	train/11/other.wav	228
train/12/mixture.wav	train/12/vocals.wav	train/12/drums.wav	train/12/bass.wav	train/12/other.wav	205
train/13/mixture.wav	train/13/vocals.wav	train/13/drums.wav	train/13/bass.wav	train/13/other.wav	214
train/14/mixture.wav	train/14/vocals.wav	train/14/drums.wav	train/14/bass.wav	train/14/other.wav	243

그림 5. train_csv, validation_csv 파일의 구성 정보
Fig. 5. Configuration information in the train_csv, validation_csv files

구하여 Note number로 변환한 PitchLabel 데이터가 요구된다. Pitch란 음의 높낮이를 나타내는 속성으로, 음악을 구성하는 주파수의 높낮이를 의미한다. Note number는 음악에서 각 음의 위치를 숫자로 표현하는 방법으로, 주로 MIDI와 같은 음악 기기와 소프트웨어에서 사용된다.

보컬 분리 음원에서 Pitch를 추출하기 위해 본 논문에서는 YIN 알고리즘[15]을 이용하여 Pitch Extractor를 구현하였다. 해당 Extractor를 이용해 Pitch를 추출하고, 추출한 값을 pv 확장자를 가진 파일로 저장한다. 이후 pv 파일과 원본 wav 파일을 이용하여 feature를 생성한다. 드럼 채보 모델의 경우 원본 midi 파일에서 드럼에 해당하는 리듬 트랙을 추출한 뒤, 추출한 리듬 트랙 midi 파일과 원본 wav 파일을 이용해 feature를 생성한다. 코드 채보 모델의 경우 feature 생성을 위해서는 세 개의 파일이 요구된다. 그 중 bothchroma.csv 파일과 majmin.lab 파일은 학습에 사용할 데이터를 이용하여 일련의 과정을 거쳐야 한다.

bothchroma.csv 파일의 생성을 위해서 원본 wav 파일에서 Chordino Vamp Plugin과 Sonic Visualiser를 이용하여 wav 파일에서 Chorma로 transform을 수행하고, 그 결과를 csv 파일로 저장하였다. Chroma란 음악의 음높이 정보를 표현하기 위한 방법 중 하나이며, 음악을 12개의 음계로 분할하여 각 음계에 해당하는 주파수 대역의 에너지를 나타내는 방식으로 표현된다. Chroma Transform은 시계열 형태로 된 음악 신호에서 주파수 대역을 Chroma 표현으로 변환하는 과정을 의미한다. 이를 위해 시계열 신호를 작은 단위로 나누고, 각 단위에서의 주파수 구성을 분석하여 음의 높이에 따른 강도를 계산한다. 그림 6은 Chroma Transform을 수행한 결과 이미지로, 각 픽셀의 밝기는 해당 음계의 주파수 대역 에너지를 나타내며 특정 영역의 밝기가 높을수록 해당 음계의 주파수 대역이 원래 음악에서 더

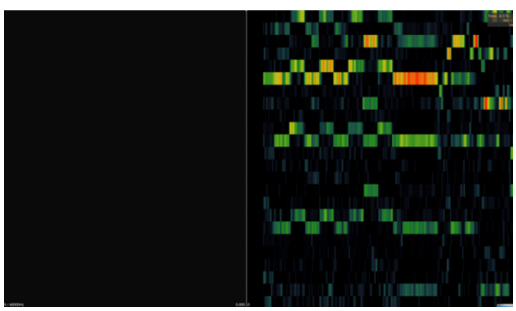

그림 6. Chroma transform 수행 결과
Fig. 6. Result of performing a Chroma transform

0	0	0	0	0	0	0	0
0.042667	0	0	0	0	0	0	0
0.085333	0	0	0	0	0	0	0
0.128	0	0	0	0	0	0	0
0.170667	0	0	0	0	0	0	0
0.213333	0	0	0	0	0	0	0
0.256	0	0	0	0	0	0	0
0.298667	0	0	0	0	0	0	0
0.341333	0	0	0	0	0	0	0
0.384	0	0	0	0	0	0	0
0.426667	0	0	0	0	0	0	0
0.469333	0	0	0	0	0	0	0
0.512	0	0	0	0	0	0	0
0.554667	0	0	0	0	0	0	0
0.597333	0	0	0	0	0	0	0
0.64	0	0	0	0	0	0	0

그림 7. bothchroma.csv 파일의 구성 정보
Fig. 7. Configuration information in the bothchroma.csv

0	0.9090914999999999	G#:min
0.9090914999999999	1.8181829999999999	Eb:min
1.8181829999999999	3.030305	Eb:min
3.030305	3.3333354999999996	C#:min
3.3333354999999996	3.6363659999999998	C:min
3.6363659999999998	4.5454574999999995	Bb:min
4.5454574999999995	5.454548999999999	C#:min
5.454548999999999	6.666670999999999	G#:min
6.666670999999999	6.969701499999999	F:min
6.969701499999999	7.2727319999999995	G:min
7.2727319999999995	8.03030825	C#:maj
8.03030825	8.787884499999999	Eb:min
8.787884499999999	9.545460749999998	Bb:min
9.545460749999998	10.303036999999998	Eb:min
10.303036999999998	11.212128499999999	G#:min
11.212128499999999	12.12122	G#:min

그림 8. majmin.lab 파일의 구성 정보
Fig. 8. Configuration information in the majmin.lab

두드러져 들리거나 더 강조되는 것을 의미한다. 그림 7은 bothchroma.csv 파일의 구성을 보여준다.

majmin.lab 파일은 그림 8에서 볼 수 있듯, 곡의 onset time, offset time, chord 정보가 순서대로 적혀있는 파일이다. onset은 음악에서 소리가 시작되는 시점, 즉 음표 및 소리의 시작점을 말하며, offset은 음악에서 소리가 끝나는 시점, 즉 음표 혹은 소리가 끝나는 지점을 의미한다.

4.3 모델 학습 및 테스트

Spleeter 학습 시 설정한 파라미터는 표 4와 같으며, Spleeter의 평가 지표로는 Signal to Distortion Ratio(SDR)를 선정하였으며 SDR의 계산식은 식 (1)과 같다. 해당 식에서 s_target은 분리를 원하는 대상 음원

의 절댓값을, e_interf는 분리된 음원에 남아있는 간섭 신호들을, e_artif는 음원이 분리되는 과정에서 발생한 신호 자체의 결함을 의미한다. 그림 9는 Spleeter 모델의 학습 결과 로그를 나타내며, SDR 5.439를 달성하였다.

Omnizart 학습 시 설정한 파라미터는 표 5와 같으며, 평가 지표로는 F1-Score를 선정하였다. F1-Score의 계산식은 식 (2)와 같다. 표 6은 Omnizart Vocal, Drum, Chord 모델의 학습 결과를 보여준다.

$$SDR := 10\log_{10}\left(\frac{\| s_{target} \|^2}{\| e_{interf} + e_{noise} + e_{artif} \|^2}\right) \quad (1)$$

표 4. Spleeter 모델의 파라미터 설정
Table 4. Setting parameters for Spleeter model

Parameters	values
batch_size	4
learning rate	0.0001
max_steps	200000

표 5. Omnizart 모델의 파라미터 설정
Table 5. Setting parameters for Omnizart models

type	epochs	batch_size	val_batch_size
Vocal	10	16	16
Drum	10	16	16
Chord	10	64	64

표 6. Omnizart 모델의 학습 결과
Table 6. Training results for Omnizart models

type	F1-score
Vocal	0.853
Drum	0.695
Chord	0.392

```
--------------------------------
Spleeter SDR
Start Time : 2023-07-05 03:16:33.473782
End Time : 2023-07-05 06:25:30.057728
SDR Mean : 5.439
--------------------------------
```

그림 9. Spleeter 모델 학습 결과 로그
Fig. 9. Spleeter model training result log

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} \quad (2)$$

학습 결과 Omnizart Vocal이 0.853으로 가장 높은 F1-score 결과 값을 얻었으며, Omnizart Chord의 경우 Vocal, Drum에 비해 낮은 수치를 얻었다.

4.4 유사도 측정

MelodyShape을 활용하여 유사도를 측정한 결과는 표 7과, GrooveToolBox를 활용하여 드럼의 유사도를 측정한 결과는 표 8과 같다.

MelodyShape은 비교하는 두 곡의 멜로디가 유사하다면 결과값이 높게 산출되며, 그렇지 않을 경우 낮은 결과값을 산출하게 된다. MelodyShape을 통한 유사도 측정의 정확성을 확인하기 위해 'Dance_00936_Org'와 해당 곡에서 장르를 변경한 'Dance_00936_Tempo_B' 및 이와 유사하지 않은 3개의 다른 곡을 선정하여 유사도 측정을 진행하였다. 측정 결과, 멜로디가 동일한 곡인 'Dance_00936_Org'와 'Dance_00936_Tempo_B' 사이의 유사도가 99.3369로 매우 높게 측정된 것을 확인할 수 있었다. 또한 다른 곡 중에서도 일부 유사도가 측정된 것을 확인할 수 있었다.

힙합 장르의 곡을 GrooveToolBox로 유사도를 측정한 결과, 모든 장르에서 전체적으로 93-85 사이의 높은 결과값을 얻는 것을 확인할 수 있었다. 하지만 음악의 실질적 유사성을 판단하는 것은 멜로디가 중심이 되기에, MelodyShape을 통한 유사도 측정을 우선으로 진행한 뒤, 그 중 높은 유사도를 가진 곡들을 대상으로 GrooveToolBox를 통해 리듬의 유사도를 측정하는 방

표 7. MelodyShape의 유사도 측정 결과
Table 7. Similarity measure results for MelodyShape

	Dance_00936_Org
Dance_00936_Tempo_B	99.3369
Dance_00904_Org	0.3003
Dance_00903_Genre_A	0.0852
Ballade_00811_Timbre_D	0

표 8. GrooveToolBox의 유사도 측정 결과
Table 8. Similarity measure results for GrooveToolBox

	Hiphop_00877_Org
Dance_00902_Timbre_D	93.591542
Trot_00891_Org	91.578642
Ballade_00955_Tempo_D	90.46484
Hiphop_00941_Genre_A	85.924574

식으로 활용하는 것이 적합하다고 판단된다.

V. 결 론

본 논문에서는 이전에 제안했던 음악 유사성 분석 연구를 기반으로 실제 음악 유사성 분석 시스템을 구현하기 위해 각 단계에서 오픈소스 인공지능 모델과 알고리즘을 조사 및 선정하였다. 선정한 인공지능 모델의 학습을 위해 음악 데이터를 수집하여 데이터셋을 구축하고, 사용한 각 모델의 학습 데이터 구조에 맞게 전처리를 진행하고 학습을 진행하였다.

여러 트랙이 혼합되어 있는 기존의 혼합 음악 데이터에서 Spleeter 모델을 이용해 보컬, 드럼, 베이스(화성) 트랙을 추출해 각각의 분리 음원을 생성하고, 분리된 음원을 Omnizart Vocal, Drum, Chord 모델의 입력 데이터로 사용하여 자동 채보를 통해 최종적으로 MIDI 형식의 파일을 생성하고, 해당 MIDI 파일을 이용하여 유사성 분석을 진행한다. 보컬, 화성 트랙은 MelodyShape을, 드럼 트랙은 GrooveToolBox를 이용해 유사성을 분석하였다. 이는 혼합 음악 데이터의 멜로디를 중심으로 하는 기존 유사성 분석 방식과는 달리, 3개의 트랙으로 음악 데이터를 분리하여 독립적이고 자세한 분석을 통해 다양한 음악적 특성을 고려하여 더 정확하고 포괄적인 유사도 평가가 가능하며, 다양한 음악 장르 간의 유사성을 평가하는데 있어서도 더 나은 분석 시스템의 제공을 기대할 수 있다. 또한 해당 분석 시스템을 이용하여 기존에 발매된 음원 간의 유사성 분석뿐만이 아닌, 새로운 음원을 제작하고 있는 단계에서도 기존 음원과의 표절 문제를 피하기 위한 수단으로서의 활용이 가능할 것으로 예상된다.

본 시스템의 음원 분리와 자동 채보 단계에서는 인공지능을 기반으로, 유사도 측정 단계는 알고리즘을 통해 구현하였는데, 추후 인공지능 기반의 유사도 측정 관련 연구를 진행하고 현재의 유사성 분석 시스템에 도입해 음악 유사성 분석의 전 과정에서 인공지능이 기반이 되는, 인공지능 기반의 음악 유사성 분석 시스템을 구현할 계획이다.

References

[1] Y. Gu and J. Lee, “Design of an analysis system for music similarity detection,” in *Proc. KICS ICC 2023*, pp. 404-405, Jeju Island, Korea, Jun. 2023.

[2] F. Stöter, et al., “Open-unmix-a reference implementation for music source separation,” *J. Open Source Softw.*, vol. 4, no. 41, 1667, 2019. (https://doi.org/10.21105/joss.01667)

[3] PyTorch, *Open-Unmix* (Accessed Jul. 19, 2023) (URL: https://pytorch.kr/hub/sigsep_open-unmix- pytorch_umx/)

[4] J. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, “MT3: Multi-task multitrack music transcription,” *arXiv preprint arXiv:2111.03017*, 2021. (https://doi.org/10.48550/arXiv.2111.03017)

[5] Magenta, *Transcription with Transformers* (Accessed Jul. 20, 2023) (URL: https://magenta.tensorflow.org/transcription-with-transformers)

[6] Albincorreya, *ChromaCoverId*, Retrieved Jul. 23, 2023, from https://github.com/albincorreya/ChromaCoverId

[7] J. Serra, et al., “Cross recurrence quantification for cover song identification,” *New J. Physics*, vol. 11, no. 9, 093017, 2009.

[8] N. Chen, et al., “Fusing similarity functions for cover song identification,” *Multimedia Tools and Appl.*, vol. 77, pp. 2629-2652, 2018.

[9] T. Huang, et al., “MidiFind: Fast and effective similarity searching in large MIDI databases,” in *Proc. 10th Int. Symp. Computer Music Multidisciplinary Res.*, vol. 16, pp. 209-224, Marseille, France, 2013.

[10] R. Castellon, et al., “Codified audio language modeling learns useful representations for music information retrieval,” *arXiv preprint arXiv:10705677*, 2021.

[11] R. Hennequin, et al., “Spleeter: A fast and efficient music source separation tool with pre-trained models,” *J. Open Source Softw.*, vol. 5, no. 50, 2154, 2020. (https://doi.org/10.21105/joss.02154)

[12] Y. Wu, et al., “Omnizart: A general toolbox for automatic music transcription,” *J. Open Source Softw.*, vol. 6, no. 68, 3391, 2021. (https://doi.org/10.21105/joss.03391)

[13] J. Urbano, et al., “Melodic similarity through shape similarity,” in *CMMR 2010*, pp.

338-355, Málaga, Spain, Jun. 2010.

[14] Fredbru, *GrooveToolbox*, Retrieved Jul. 23, 2023, from https://github.com/fredbru/GrooveToolbox

[15] A. De Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The J. Acoustical Soc. Am.*, vol. 111, no. 4, pp. 1917-1930, 2002. (https://doi.org/10.1121/1.14 58024)

구 연 우 (Yeonwoo Gu)

2020년 3월~현재 : 덕성여자대학교 소프트웨어전공 학사과정

<관심분야> Machine Learning, GAN, Neural Network

[ORCID:0009-0003-3628-5933]

이 재 호 (Jaeho Lee)

2005년 : 고려대학교 전자컴퓨터공학과 석사

2008년~2013년 : 고려대학교 전기전자전파공학과 박사

2013년~2015년 : LG전자 차세대표준연구소 선임연구원

2015년~2019년 : 서원대학교 정보통신공학과 조교수

2020년~현재 : 덕성여자대학교 소프트웨어전공 조교수

<관심분야> WPAN, MAC, Bluetooth, Wi-Fi, Localization, NLP, Machine Learning

[ORCID:0000-0003-0455-9939]