Hand Craft Model

Yeong-Min Ko & Min-Seo Park

2023.12.27~2023.12.28

I . Introduction to our algorithms

i. Evaluation Metrics
ii. How to implement

I. Introduction to our Algorithms

i. Evaluation Metrics

	평가 기준	매트릭	
1	음정	note	
2	셈여림	decibel	
3	셈여림의 변화	change of decibel	
4	빠르기	speed	
5	빠르기의 변화	change of speed	일부 구현 - 추가 고민 필요
6	붙임줄, 스타카토, 테누토, 늘임표	duration of note	
7	악센트	decibel	
8	옥타브	note	
9	꾸밈음, 반복 기호	note	
10	리듬	?	
11	페달링	pedaling	- 인풋과 정답 데이터의 각 페달 상태를 배열화 시켜 비교하여 일치 정도를 계산하도록 구현했는데 코드 검토 필요
12	음의 길이	duration of note	일부 구현 - 추가 고민 필요

공식화 가능한 기준

I. Introduction to our Algorithms

ii. How to implement - 음정(note)

- 기존에 input과 target midi 데이터를 주파수로 변환해서 특정 구간마다 각각 비교하고자 했으나 노트를 추출해서 비교하는 것으로 수정
- **음정 정확도 계산**
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 두 파일의 notes를 읽어와 set() 함수를 통해 중복된 노트를 제거
    - 3) 두 파일의 notes 중 교집합(intersection)을 구해 모두 등장하는 노트를 구함
    - 4) 공통된 노트의 개수를 input midi file의 노트 개수로 나누어 백분율로 표현
  - 핵심 코드
    - pitch_accuracy = (len(common_notes) / len(input_notes)) * 100

- 이후 음정 정확도에 기반하여 사용자의 수준(level)을 계산
  - 방법
    - 입력된 정확도를 10% 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(accuracy / 10)))

I. Introduction to our Algorithms

ii. How to implement - 빠르기(speed) - 추가 조사 필요

- **빠르기 유사도 계산**
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 두 파일의 각 트랙별 tempo를 불러와 템포의 평균을 계산
    - 3) 임계값(thresholds)를 두고 임계값 이내이면 100을 부여, 그렇지 않으면 템포 차이에 따라 가중치를 부여하여 점수를 계산
      - score = max(0, 100 - min(3, excess_difference) * (tempo_difference - threshold))
      - 구체적인 가중치 계산 방법은 아직 프로토타입 상태
  - 핵심 코드
    - excess_difference = max(0, tempo_difference - threshold) # 임계값을 초과한 템포 차이
    - if tempo_difference <= threshold:
    - score = 100 # threshold 이하의 템포 차이는 최고 점수 100을 부여
    - else:
    - # threshold를 초과하는 경우 템포 차이에 따라 점수 부여
    - # 차후에 가중치 생각해 볼 필요 있음 - 현재는 임시 로직
    - score = max(0, 100 - min(3, excess_difference) * (tempo_difference - threshold))

- 이후 빠르기 유사도 점수에 기반하여 사용자의 수준(leve)을 계산
  - 방법
    - 입력된 점수를 10점 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(score / 10)))

I. Introduction to our Algorithms

ii. How to implement - 빠르기의 변화(change of speed) - 추가 조사 필요

- **빠르기 변화 계산**
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 두 midi file에서 각각 템포 변화가 발생한 트랙 번호(i), 템포 변화 발생 시간의 초(time_in_seconds), 템포 변화 시작 시점의 템포(current_tempo), 끝 템포(tempo)를 추출
    - 3) 작은 값만큼 반복문을 수행하여 두 파일 간의 공통된 템포 변화 부분만을 비교
      - 두 MIDI 파일이 서로 다른 길이를 가질 수 있고, 공통된 부분만을 비교함으로써 일관성 있는 결과를 얻을 수 있기 때문
  - 핵심 코드
    - 계산 로직 아직 미구현

- 이후 빠르기 변화의 유사도 점수에 기반하여 사용자의 수준(leve)을 계산할 예정
  - 방법
    - 입력된 정확도를 10점 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(score / 10)))

I. Introduction to our Algorithms

ii. How to implement - 옥타브(octave using note)

- 옥타브 평가 점수 계산(일치도)
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 모든 트랙(track)의 모든 노트(note)에서 'note_on' 메시지를 찾아 노트의 옥타브 정보를 추출
    - 3) 각 노트의 MIDI 노트 번호를 12로 나누어 옥타브를 계산하고, 이를 리스트에 저장
      - input_octaves = [msg.note // 12 for track in input_mid.tracks for msg in track if msg.type == 'note_on']
      - target_octaves = [msg.note // 12 for track in target_mid.tracks for msg in track if msg.type == 'note_on']
    - 4) zip 함수를 사용하여 두 MIDI 파일의 옥타브 정보를 하나씩 비교하여, 일치하는 경우, octave_matching_score에 1을 더함
    - 5) octave_matching_score를 두 MIDI 파일의 전체 노트 수로 나누어 일치하는 비율 계산 후 100을 곱하여 최종 점수를 계산
  - 핵심 코드
    - # 두 곡의 옥타브 일치 여부를 평가하여 점수를 계산
    - octave_matching_score = sum(1 for input_octave, target_octave in zip(input_octaves, target_octaves) if input_octave == target_octave)
    - octave_matching_ratio = octave_matching_score / len(target_octaves) # 두 곡의 옥타브 일치 비율을 계산
    - score = octave_matching_ratio * 100 # 옥타브 일치 점수를 만점(100점) 기준으로 계산

- 이후 옥타브 평가 점수에 기반하여 사용자의 수준(leve)을 계산
  - 방법
    - 입력된 점수를 10점 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(score / 10)))

I. Introduction to our Algorithms

ii. How to implement - 페달링(pedaling) - 코드 검토 필요

- **페달링 계산**
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 두 파일의 각 페달 상태를 추출
    - 3) 입력 파일의 페달 상태와 대상 파일의 페달 상태(on | off)를 비교하여 일치하는 부분의 수 계산
      - matching_pedals = sum(1 for p1, p2 in zip(input_pedals, target_pedals) if p1 == p2)
    - 4) 전체 페달 상태 중 일치하는 페달 상태의 비율을 계산 후 페달링 점수 계산
  - 핵심 코드
    - # 두 곡의 페달링 평가
    - if len(input_pedals) == 0 or len(target_pedals) == 0:
    - return 0.0 # 하나라도 곡이 페달 정보를 갖고 있지 않으면 평가 불가
    - # 페달링이 일치하는 부분의 수를 계산
    - matching_pedals = sum(1 for p1, p2 in zip(input_pedals, target_pedals) if p1 == p2)
    - pedal_matching_ratio = matching_pedals / len(input_pedals) # 페달링 일치 비율을 계산

- 이후 페달링 점수에 기반하여 사용자의 수준(leve)을 계산
  - 방법
    - 입력된 점수를 10점 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(score / 10)))

I. Introduction to our Algorithms

ii. How to implement - 음의 지속시간(duration of note) - 추가 조사 필요

- 음의 지속시간 유사도 계산
  - 방법
    - 1) input, target의 midi file 불러오기
    - 2) 두 파일의 각 트랙의 노트 이벤트를 반복하며 노트의 시작과 끝 시간을 추출하여 노트의 길이(note duration)을 리스트에 저장
    - 3) 두 파일의 음표 길이 리스트를 받아와서 길이를 일치시킴(두 파일의 음표 길이 리스트를 받아와서 길이를 일치시킴)
      - 두 파일의 재생 시간(? 총 파일의 시간) 차이로 인해 발생할 수 있는 불일치를 최소화하기 위한 목적
    - 4) 길이를 일치시킨 두 음악의 음표 길이 리스트를 받아와 각 음표의 길이 차이의 평균을 계산하여 유사성 점수 계산
      - 유사성 임계값보다 작을 때는 100점을 주고, 그렇지 않을 경우 평균 길이 차이를 100으로 나눈 나머지를 점수로 반환
  - 핵심 코드
    - # 노트 길이 차이의 절대값을 계산하여 유사도 점수 산출
    - absolute_differences = [abs(a - b) for a, b in zip(note_durations_1, note_durations_2)]
    - average_difference = sum(absolute_differences) / len(absolute_differences)
    - if average_difference < threshold:
    - return 100, note_durations_1, note_durations_2
    - else:
    - return average_difference % 100, note_durations_1, note_durations_2

- 이후 음의 지속시간 유사도 점수에 기반하여 사용자의 수준(leve)을 계산
  - 방법
    - 입력된 점수를 10점 단위로 나누어 레벨 계산
    - 최대 레벨은 10으로 제한, 반올림 적용
  - 핵심 코드
    - level = int(min(10, round(score / 10)))

THANK YOU

컴퓨터공학과 3학년 고영민

2023.12.27~2023.12.28