음정 정확도의 경우

아래 그림과 같이 input_data와 target_data간의 싱크 문제로 인해 0.1초의 오차 발생에 따라 음정 정확도를 측정할 때 대부분의 값들이 틀리게 알고리즘이 계산됨. 따라서 낮은 정확도를 출력 하는 것으로 추측됩니다.

csv 파일 : 타깃 데이터

sec	msg_type	channel	note	velocity	dynamic	accent	count	main_vol	depth	pedal	pan
0	['note_on']	[0]	[60]	[61]	p	0	1	[100, 100,	[127, 127]	127	[64, 6
0.1	[]	[]	[]	[]		0	0	0	0	0	
0.2	['note_on']	[0]	[64]	[57]	p	0	1	0	0	0	
0.3	[]	[]	[]	[]		0	0	0	0	0	
0.4	['note_on']	[0]	[67]	[56]	p	0	1	0	0	0	
0.5	[]	[]	[]	[]		0	0	0	0	0	
0.6	['note_off',	[0, 0]	[67, 72]	[64, 60]	mp	0	2	0	0	0	
0.7	[]	[]	[]	[]		0	0	0	0	0	
0.8	['note_off',	[0, 0]	[72, 76]	[64, 63]	mp	0	2	0	0	0	
0.9	[]	[]	[]	[]		0	0	0	0	0	
1	['note_off',	[0, 0]	[76, 67]	[64, 50]	p	0	2	0	0	0	
1.1	[]	[]	[]	[]		0	0	0	0	0	
1.2	[]	[]	[]	[]		0	0	0	0	0	
1.3	['note_off',	[0, 0]	[67, 72]	[64, 47]	p	0	2	0	0	0	
1.4	[]	[]	[]	[]		0	0	0	0	0	
1.5	['note_off',	[0, 0]	[72, 76]	[64, 47]	p	0	2	0	0	0	

csv 파일 : 인풋 데이터

sec	msg_type	channel	note	velocity	dynamic	accent	count	main_vol	depth	pedal	pan
0	[]	[]	[]	[]	ppp	0	0	100	0	0	0
0.1	['note_on',	[0, 0]	[60, 60]	[21, 21]	ppp	0	2	0	0	0	0
0.2	[]	[]	[]	[]		0	0	0	0	0	0
0.3	['note_on',	[0, 0]	[64, 64]	[32, 32]	ppp	0	2	0	0	0	0
0.4	[]	[]	[]	[]		0	0	0	0	0	0
0.5	['note_on',	[0, 0]	[67, 67]	[34, 34]	ppp	0	2	0	0	0	0
0.6	[]	[]	[]	[]		0	0	0	0	0	0
0.7	['note_on',	[0, 0, 0, 0]	[72, 72, 67	[34, 34, 0,	ppp	0	4	0	0	0	0
0.8	[]	[]	[]	[]		0	0	0	0	0	0
0.9	['note_on',	[0, 0, 0, 0]	[76, 76, 72	[46, 46, 0,	ppp	0	4	0	0	0	0
1	[]	[]	[]	[]		0	0	0	0	0	0
1.1	['note_on',	[0, 0]	[67, 67]	[38, 38]	pp	0	2	0	0	0	0
1.2	['note_off',	[0, 0, 0, 0]	[60, 60, 76	[0, 0, 0, 0]	ppp	0	4	0	0	0	0
1.3	['note_on',	[0, 0, 0, 0,	[72, 72, 64	[36, 36, 0,	ppp	0	6	0	0	0	0
1.4	[]	[]	[]	[]		0	0	0	0	0	0
1.5	['note_on',	[0, 0, 0, 0]	[76, 76, 72	[46, 46, 0,	ppp	0	4	0	0	0	0
1.6	[]	[]	[]	[]		0	0	0	0	0	0
1.7	['note_on',	[0, 0, 0, 0]	[60, 60, 76	[35, 35, 0,	ppp	0	4	0	0	[126, 126,	0
1.8	[]	[]	[]	[]	ppp	0	0	0	0	[80, 80, 18	0
1.9	['note_on',	[0, 0]	[64, 64]	[31, 31]	ppp	0	2	0	0	0	0
2	[]	[]	[]	[]	ppp	0	0	0	0	[4, 4, 8, 8]	

알고리즘 설명

입력 데이터와 정답 데이터의 Notes 를 비교하여, 현재 노트가 거의 일치하면 1.5점을 부여, 약간의 차이는 1점 부여, 조금 더 많이 차이나면 0.5점 부여

알고리즘 과정 출력

```
0행의 input_data: []
0행의 target_data: [60]
1행의 input_data: [60, 60]
1행의 target_data: []
2행의 input_data: []
2행의 target_data: [64]
3행의 input_data: [64, 64]
3행의 target_data: []
4행의 input_data: []
4행의 target_data: [67]
5행의 input_data: [67, 67]
5행의 target_data: []
6행의 input_data: []
6행의 target_data: [67, 72]
7행의 input_data: [72, 72, 67, 67]
7행의 target_data: []
8행의 input_data: []
8행의 target_data: [72, 76]
```

샘여림 유사도의 경우

알고리즘 분석 결과 실제로 입력 데이터와 출력 데이터 간의 샘여림 기호의 수의 차이가 큽니다. (정답 데이터의 경우 실제로 사람이 친것인지 아닌지 데이터를 모르기 때문에 임의로 사람이 만들었을 경우 정확한 샘여림 측정을 하기 힘들어보임.)

알고리즘 출력 결과

```
{'ppp': 479, '': 441, 'pp': 234, 'p': 49, 'mp': 3}
{'': 634, 'p': 426, 'mp': 114, 'ppp': 20, 'pp': 20}
{'', 'mp', 'ppp', 'pp', 'p'}
샘여림 기호 중 큰 수의 값:  [634, 114, 479, 234, 426]
샘여림 기호의 차이 값:  [193, 111, 459, 214, 377]
1887
1354
셈여림 유사도: 28.25%
```

입력 데이터

sec	msg_type	channel	note	velocity	dynamic
0	['note_on']	[0]	[60]	[61]	p
0.1	[]	[]	[]	[]	
0.2	['note_on']	[0]	[64]	[57]	p
0.3	[]	[]	[]	[]	
0.4	['note_on']	[0]	[67]	[56]	p
0.5	[]	[]	[]	[]	
0.6	['note_off',	[0, 0]	[67, 72]	[64, 60]	mp
0.7	[]	[]	[]	[]	
0.8	['note_off',	[0, 0]	[72, 76]	[64, 63]	mp
0.9	[]	[]	[]	[]	
1	['note_off',	[0, 0]	[76, 67]	[64, 50]	p
1.1	[]	[]	[]	[]	
1.2	[]	[]	[]	[]	
1.3	['note_off',	[0, 0]	[67, 72]	[64, 47]	p
1.4	[]	[]	[]	[]	
1.5	['note_off',	[0, 0]	[72, 76]	[64, 47]	p
1.6	[]	[]	[]	[]	
1.7	['note_off',	[0, 0, 0, 0]	[60, 64, 76	[64, 64, 64	mp
1.8	[]	[]	[]	[]	
1.9	['note_on']	[0]	[64]	[57]	p
2	[]	[]	[]	[]	

출력 데이터

	A	B	C	D	E	F
1	sec	msg_type	channel	note	velocity	dynamic
2	0	[]	[]	[]	[]	ppp
3	0.1	['note_on',	[0, 0]	[60, 60]	[21, 21]	ppp
4	0.2	[]	[]	[]	[]	
5	0.3	['note_on',	[0, 0]	[64, 64]	[32, 32]	ppp
6	0.4	[]	[]	[]	[]	
7	0.5	['note_on',	[0, 0]	[67, 67]	[34, 34]	ppp
8	0.6	[]	[]	[]	[]	
9	0.7	['note_on',	[0, 0, 0, 0]	[72, 72, 67	[34, 34, 0,	ppp
10	0.8	[]	[]	[]	[]	
11	0.9	['note_on',	[0, 0, 0, 0]	[76, 76, 72	[46, 46, 0,	ppp
12	1	[]	[]	[]	[]	
13	1.1	['note_on',	[0, 0]	[67, 67]	[38, 38]	pp
14	1.2	['note_off',	[0, 0, 0, 0]	[60, 60, 76	[0, 0, 0, 0]	ppp
15	1.3	['note_on',	[0, 0, 0, 0,	[72, 72, 64	[36, 36, 0,	ppp
16	1.4	[]	[]	[]	[]	
17	1.5	['note_on',	[0, 0, 0, 0]	[76, 76, 72	[46, 46, 0,	ppp
18	1.6	[]	[]	[]	[]	
19	1.7	['note_on',	[0, 0, 0, 0]	[60, 60, 76	[35, 35, 0,	ppp
20	1.8	[]	[]	[]	[]	ppp
21	1.9	['note_on',	[0, 0]	[64, 64]	[31, 31]	ppp
22	2	[]	[]	[]	[]	ppp

알고리즘 설명

모든 행에서의 샘여림 기호 값의 합을 구해서, 모두 다 틀렸을 경우에서 맞은 부분 만큼만 % 비율로 나타낸다.

샘여림의 변화 일관성의 경우

샘여림 변화 일관성의 경우도 마찬가지로 기존의 정답 데이터와 입력 데이터의 샘여림이 많이 다르므로, 샘여림 변화 패턴도 다르다 따라서 낮은 정확도를 보임.

알고리즘 설명

샘여림 변화하는 구간의 수를 세어 공통되는 패턴의 수를 전체 패턴의 수로 나누어 준다.

아래는 'ppp'에서 ' ' 샘여림 없음으로 넘어가는 것 행의 해턴이 237개 존재하는 것을 의미

{('ppp', ''): 237, (

알고리즘 코드 출력 결과

```
{('ppp', ''): 237, ('', 'ppp'): 188, ('', 'pp'): 154, ('ppp', 'ppp'): 147, ('pp', 'ppp'): 119, ('p
p', ''): 115, ('ppp', 'pp'): 79, ('', ''): 63, ('', 'p'): 34, ('p', ''): 24, ('p', 'ppp'): 23, ('pp
p', 'p'): 14, ('ppp', 'mp'): 2, ('mp', ''): 2, ('p', 'p'): 1, ('p', 'pp'): 1, ('', 'mp'): 1, ('mp',
'ppp'): 1, ('', nan): 1}
{('p', ''): 418, ('', 'p'): 408, ('', 'mp'): 111, ('', ''): 100, ('mp', ''): 90, ('pp', ''): 20,
('', 'pp'): 11, ('ppp', 'p'): 11, ('mp', 'ppp'): 8, ('mp', 'pp'): 8, ('p', 'ppp'): 7, ('mp', 'p'):
6, ('ppp', ''): 6, ('', 'ppp'): 3, ('mp', 'mp'): 2, ('ppp', 'ppp'): 2, ('p', 'mp'): 1, ('ppp', 'p
p'): 1, ('', nan): 1}
{('pp', 'ppp'): 0, ('', 'p'): 34, ('p', 'p'): 0, ('ppp', 'pp'): 1, ('mp', 'mp'): 0, ('ppp', ''): 6,
('p', ''): 24, ('mp', ''): 2, ('p', 'ppp'): 7, ('pp', ''): 20, ('', 'ppp'): 3, ('ppp', 'mp'): 0,
('', ''): 63, ('', nan): 1, ('', 'pp'): 11, ('ppp', 'p'): 11, ('', 'mp'): 1, ('p', 'mp'): 0, ('mp',
'pp'): 0, ('mp', 'p'): 0, ('ppp', 'ppp'): 2, ('p', 'pp'): 0, ('mp', 'ppp'): 1}
187
1206
쌤여림 변화 일관성: 15.51%
```

악센트 정확도의 경우

악센트 정확도의 경우 데이터 값 자체에 악센트 관련한 값이 모두 0으로 표기되어 연산이 불가능한 상태 따라서 악센트가 없으므로 0으로 표기, -> 악센트 없음으로 표기되어야 할 것으로 보임.

코드 출력 결과

```
input_data 악센트 발생 구역 []
target_data 악센트 발생 구역 []

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~₩AppData₩Local₩Temp₩ipykernel_10200₩2161185863.py in <module>
     50     return f'{accent_score:.2f}%'
     51
---> 52 accent_accuracy = calculate_accent_accuracy(input_msg_info, target_msg_info)
     53 print(f'악센트 정확도: {accent_accuracy}')

~₩AppData₩Local₩Temp₩ipykernel_10200₩2161185863.py in calculate_accent_accuracy(input_data, target_data)
     47             accent_score += 1
     48
---> 49     accent_score = int(accent_score  / len(target_time_range) * 100)
     50     return f'{accent_score:.2f}%'
     51

ZeroDivisionError: division by zero
```

옥타브 유사도의 경우

옥타브 유사도의 경우도 마찬가지로 싱크 문제가 있어
보임 -> 0.1초 차이로 노트의 값이 틀어졌을 경우 알고
리즘 계산과정에서 낮은 정확도를 얻을 수 밖에 없음.

훈련 데이터

sec	msg_type	channel	note	velocity	dynamic	accent	count	main_vol	depth	pedal	pan
0	['note_on']	[0]	[60]	[61]	p	0	1	[100, 100,	[127, 127]	127	[64, 64]
0.1	[]	[]	[]	[]		0	0	0	0	0	0
0.2	['note_on']	[0]	[64]	[57]	p	0	1	0	0	0	0
0.3	[]	[]	[]	[]		0	0	0	0	0	0
0.4	['note_on']	[0]	[67]	[56]	p	0	1	0	0	0	0

입력 데이터

sec	msg_type	channel	note	velocity	dynamic	accent	count	main_vol	depth	pedal	pan
0	[]	[]	[]	[]	ppp	0	0	100	0	0	0
0.1	['note_on',	[0, 0]	[60, 60]	[21, 21]	ppp	0	2	0	0	0	0
0.2	[]	[]	[]	[]		0	0	0	0	0	0
0.3	['note_on',	[0, 0]	[64, 64]	[32, 32]	ppp	0	2	0	0	0	0
0.4	[]	[]	[]	[]		0	0	0	0	0	0

코드 출력 결과

```
0행
[]
[1.0]
거리:  1.0
유사도 값:  0.5
================================================
1행
[1.0]
[]
거리:  1.0
유사도 값:  0.5
================================================
2행
[]
[1.0]
거리:  1.0
유사도 값:  0.5
================================================
```

패달 데이터의 경우

코드 출력 결과를 보면 인풋 데이터, 타깃 데이터 추출 과정에서 문자, 숫자의 추출이 잘못된것으로 보임.

(0, 0) , (0, 0) 은 공통 패턴인데 공통 패턴 딕셔너리에 추가가 안되어 있음.

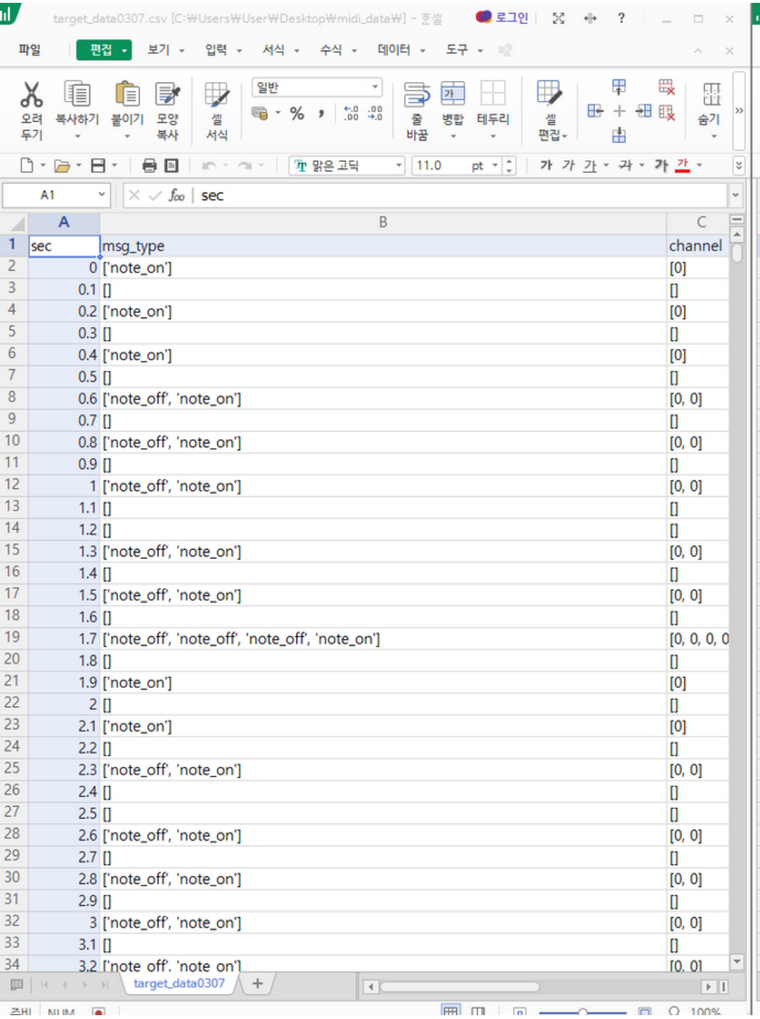
코드 출력 결과

```
인풋 데이터: {('0', '0'): 1199, ('0', '[126, 126, 106, 106]'): 1, ('[126, 126, 106, 106]', '[80, 80, 18, 18, 0, 0]'): 1,
('[80, 80, 18, 18, 0, 0]', '0'): 1, ('0', '[4, 4, 8, 8]'): 1, ('[4, 4, 8, 8]', '[6, 6, 0, 0]'): 1, ('[6, 6, 0, 0]', '0'): 1,
('0', nan): 1}
타깃 데이터: {(0, 0.0): 1144, (127, 0.0): 35, (0, 127.0): 34, (0, nan): 1}
공통 패턴: {}
0.0
페달링 일관성: 0.00%
```

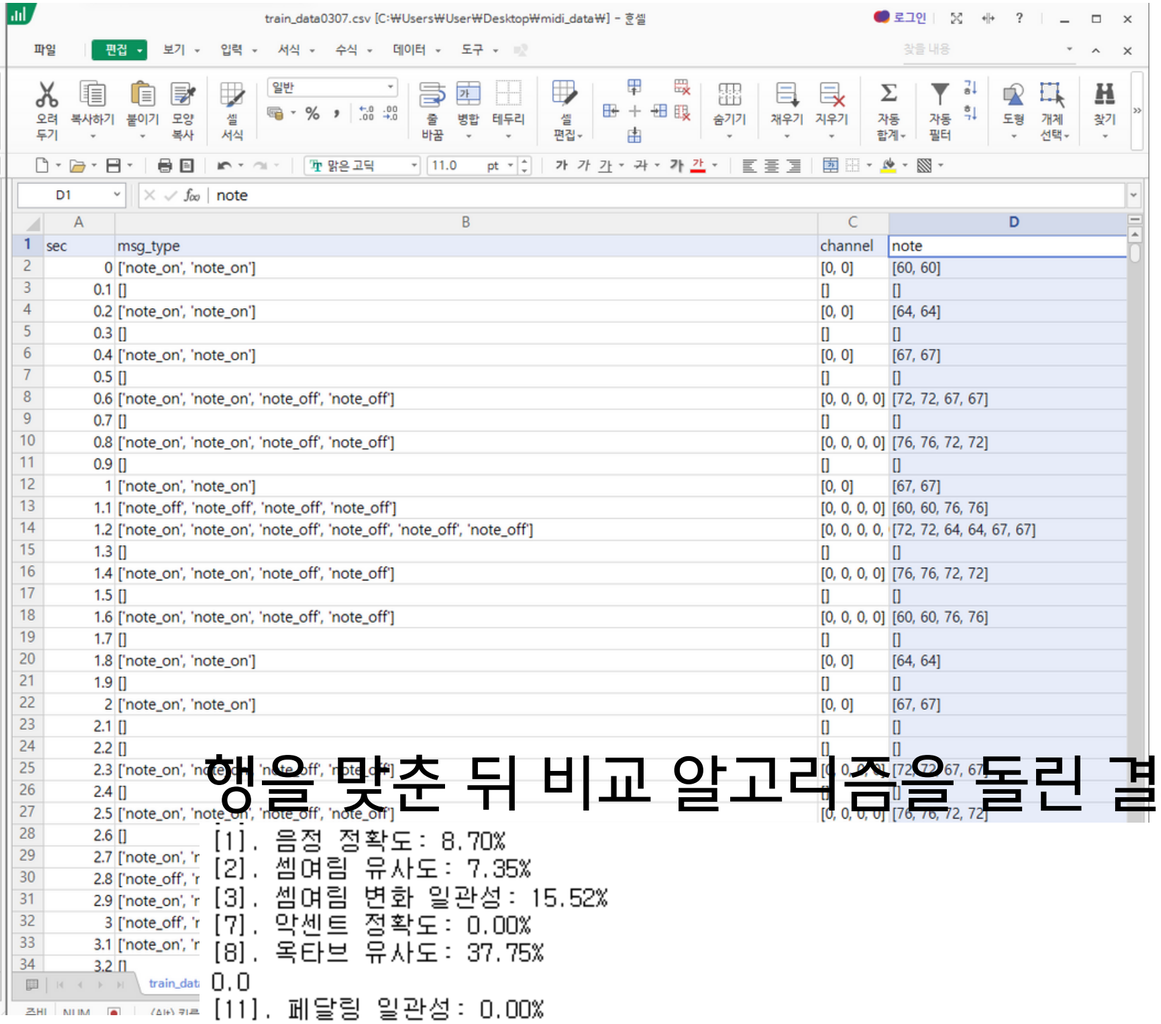
입력 데이터	출력 데이터
pedal	pedal
127	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	[126, 126,
0	[80, 80, 18,
0	0
0	[4, 4, 8, 8]
0	[6, 6, 0, 0]
0	0
0	0
0	0
0	0
0	0

추가적으로.. 훈련 데이터에 노트 값이 두개 씩 찍혀 있는데 원인을 알 수가 없어서, 원인을 찾아 보아야 할 것 같습니다.

노트 값이 두 개 씩 찍혀버리는 것 때문에 비교 알고리즘의 정확도도 낮아지는 것을 확인했습니다.



sec	msg_type	channel
0	['note_on']	[0]
0.1	[]	[]
0.2	['note_on']	[0]
0.3	[]	[]
0.4	['note_on']	[0]
0.5	[]	[]
0.6	['note_off', 'note_on']	[0, 0]
0.7	[]	[]
0.8	['note_off', 'note_on']	[0, 0]
0.9	[]	[]
1	['note_off', 'note_on']	[0, 0]
1.1	[]	[]
1.2	[]	[]
1.3	['note_off', 'note_on']	[0, 0]
1.4	[]	[]
1.5	['note_off', 'note_on']	[0, 0]
1.6	[]	[]
1.7	['note_off', 'note_off', 'note_off', 'note_on']	[0, 0, 0, 0
1.8	[]	[]
1.9	['note_on']	[0]
2	[]	[]
2.1	['note_on']	[0]
2.2	[]	[]
2.3	['note_off', 'note_on']	[0, 0]
2.4	[]	[]
2.5	[]	[]
2.6	['note_off', 'note_on']	[0, 0]
2.7	[]	[]
2.8	['note_off', 'note_on']	[0, 0]
2.9	[]	[]
3	['note_off', 'note_on']	[0, 0]
3.1	[]	[]
3.2	['note_off', 'note_on']	[0, 0]



sec	msg_type	channel	note
0	['note_on', 'note_on']	[0, 0]	[60, 60]
0.1	[]	[]	[]
0.2	['note_on', 'note_on']	[0, 0]	[64, 64]
0.3	[]	[]	[]
0.4	['note_on', 'note_on']	[0, 0]	[67, 67]
0.5	[]	[]	[]
0.6	['note_on', 'note_on', 'note_off', 'note_off']	[0, 0, 0, 0]	[72, 72, 67, 67]
0.7	[]	[]	[]
0.8	['note_on', 'note_on', 'note_off', 'note_off']	[0, 0, 0, 0]	[76, 76, 72, 72]
0.9	[]	[]	[]
1	['note_on', 'note_on']	[0, 0]	[67, 67]
1.1	['note_off', 'note_off', 'note_off', 'note_off']	[0, 0, 0, 0]	[60, 60, 76, 76]
1.2	['note_on', 'note_on', 'note_off', 'note_off', 'note_off', 'note_off']	[0, 0, 0, 0,	[72, 72, 64, 64, 67, 67]
1.3	[]	[]	[]
1.4	['note_on', 'note_on', 'note_off', 'note_off']	[0, 0, 0, 0]	[76, 76, 72, 72]
1.5	[]	[]	[]
1.6	['note_on', 'note_on', 'note_off', 'note_off']	[0, 0, 0, 0]	[60, 60, 76, 76]
1.7	[]	[]	[]
1.8	['note_on', 'note_on']	[0, 0]	[64, 64]
1.9	[]	[]	[]
2	['note_on', 'note_on']	[0, 0]	[67, 67]
2.1	[]	[]	[]
2.2	[]	[]	[]
2.3	['note_on', 'note_off', 'note_off', 'note_off']	[0, 0, 0, 0]	[72, 72, 67, 67]
2.4	[]	[]	[]
2.5	['note_on', 'note_on', 'note_off', 'note_off']	[0, 0, 0, 0]	[76, 76, 72, 72]
2.6	[]	[]	[]
2.7	['note_on', 'r	[]	[]
2.8	['note_off', 'r	[]	[]
2.9	['note_on', 'r	[]	[]
3	['note_off', 'r	[]	[]
3.1	['note_on', 'r	[]	[]
3.2	[]	[]	[]

행을 맞춘 뒤 비교 알고리즘을 돌린 결과

```
[1]. 음정 정확도: 8.70%
[2]. 셈여림 유사도: 7.35%
[3]. 셈여림 변화 일관성: 15.52%
[7]. 악센트 정확도: 0.00%
[8]. 옥타브 유사도: 37.75%
0.0
[11]. 페달링 일관성: 0.00%
```