Code setting

For `train_list` in the `train()` function, the paths to the MIDI files need to be adjusted to match the data storage location on your computer.

For example, set `train_list = glob.glob('../Pop1k7/midi_analyzed/*/*.mid').`

File and Path Descriptions

In sample_code/result/GPT-2, I stored the model's training weight files and the learning curve. Only the checkpoint for epoch 150 is included among the weight files.

The remaining weight files can be downloaded from the following URL: https://drive.google.com/drive/folders/1R1xceD3JctXjGia3s0NiXOx02728m4zr?usp=sharing.

In sample_code/testing_results, you will find 20 generated songs for each configuration, along with the H1, H4, and GS metrics for each song.

The sample_code/testing_results/prompt contains the specified MIDI files for task 2.

The sample_code/testing_results/calculate_average_metrics.py computes the average metrics for each configuration, enabling comparative analysis across different configurations.

Run sample_code/main.py

In sample_code/main.py, you can execute training for GPT-2, use the trained model to generate 20 songs, or continue a song using a given prompt. Specifically, the program sections for each task are as follows:

Train GPT-2 model: This section initializes and trains the GPT-2 model on the provided dataset.

```
## Train GPT-2
train()
```

Generate 20 songs: After training, this function utilizes the model to autonomously generate 20 complete songs.

```
## Generate 20 songs
model = 200
t = 1.2
topk = 5
save_path = f'testing results/GPT-12-{model} t-{t} topk-{topk}'
if not os.path.exists(save_path):
    os.makedirs(save_path)
i = 1
while i < 21:
    try:
        print(f'\n--------------------\nGenerate {i} song. model is {model}, t is
{t}, topk is {topk}')
        test(n_target_bar=32, temperature=t, topk=topk,
output_path=f'{save_path}/testing song {i}.mid', model_path=f'./result/GPT-
2/epoch_{model:03d}.pkl', prompt=False)
        i += 1
    except Exception as e:
        print(e)
        continue
```

Generate continuation based on a prompt: This section takes a user-provided prompt as a starting point, allowing the model to continue generating music in line with the initial input.

```
## Task 2
i = 1
while i < 4:
    try:
        temperature = 1.2
        topk = 5
        if not os.path.exists(f'testing results/prompt/GPT-12-150 t-{temperature}
topk-{topk}/'):
            os.makedirs(f'testing results/prompt/GPT-12-150 t-{temperature} topk-
{topk}/')
        print(f'Generate temperature: {temperature}, topk: {topk}, i: {i}.')
        test(n_target_bar=32, temperature=temperature, topk=topk,
output_path=f'testing results/prompt/GPT-12-150 t-{temperature} topk-
{topk}/song_{i}.mid',
             model_path=f'result/GPT-2/epoch_150.pkl', prompt=True,
prompt_path=f'../prompt_song/song_{i}.mid', init_generated_bar=8)
        i += 1
        print('Save!!!!!!!!!!!!!!')
    except Exception as e:
            print(e)
            continue
```

Each part in main.py corresponds to one of these functionalities and can be executed based on the specific task requirements.

Run eval_metrics.py

Please ensure that the path settings in test_pieces and the file storage paths are configured correctly. Adjust these paths to match your directory structure to avoid any file access or storage issues during execution.