Musicgen.py

```
#using torch
from transformers import MusicgenForConditionalGeneration;

num_s=1; #1
tokens=1024;  #256
do_s=True;

model=MusicgenForConditionalGeneration.from_pretrained("facebook/musicgen-small");

import torch;

#device=":cuda:0" if torch.cuda.is_available() else "cpu";
#device="cpu";
device="cuda:0";
model.to(device);

unconditional_inputs = model.get_unconditional_inputs(num_samples=num_s)

audio_values = model.generate(**unconditional_inputs, do_sample=do_s,
max_new_tokens=tokens);



#listen audio
from IPython.display import Audio
sampling_rate=model.config.audio_encoder.sampling_rate;
#Audio(audio_values[0].cpu().numpy(), rate=sampling_rate);

#save audio in file
import scipy;


scipy.io.wavfile.write("musicgen_out.wav", rate=sampling_rate, data=audio_values[0,
0].cpu().numpy())

audio_length_in_s = tokens / model.config.audio_encoder.frame_rate

print("audio_in_sec=",audio_length_in_s);

89238497239874982374982378947
```

musicgen2.py

```
#using torch
from transformers import MusicgenForConditionalGeneration;
import torch;
from IPython.display import Audio;
import scipy;
from os import close;
```

```
num_s=1; #1
tokens=1024;  #256
do_s=True;
num_of_data=2500;

model=MusicgenForConditionalGeneration.from_pretrained("facebook/musicgen-small");

#device=":cuda:0" if torch.cuda.is_available() else "cpu";
#device="cpu";
device="cuda:0";
model.to(device);

unconditional_inputs = model.get_unconditional_inputs(num_samples=num_s)
sampling_rate=model.config.audio_encoder.sampling_rate;


#make data
for i in range(0,num_of_data):

   audio_values = model.generate(**unconditional_inputs, do_sample=do_s,
max_new_tokens=tokens);

#save audio in file
   name_of_file=str(i);
   name_of_file+='.wav';


   scipy.io.wavfile.write(name_of_file, rate=sampling_rate, data=audio_values[0, 0].cpu().numpy());

   audio_length_in_s = tokens / model.config.audio_encoder.frame_rate

   print("audio_in_sec=",audio_length_in_s);
```

234322347802138409823094812398 4-
musicgen_FROM_txtfile.py

```
#using torch
import torch;
import scipy;
import torchaudio
from audiocraft.models import AudioGen
from audiocraft.data.audio import audio_write
from IPython.display import Audio;
import soundfile as sf;

duration=40;  # generate 5 seconds.
descriptions = ['Vivaldi, Barroacco, Violin'];
```

```
num_s=1; #1
tokens=1024;  #256
do_s=True;
num_of_data=5040;




#device=":cuda:0" if torch.cuda.is_available() else "cpu";
#device="cpu";
device="cuda:0";

model = AudioGen.get_pretrained("facebook/musicgen-small");
model.set_generation_params(duration);
#sampling_rate=model.config.audio_encoder.sampling_rate;
sampling_rate=32000;
#model.to(device);




pieces=[];
#make data
for i in range(0,num_of_data):

   audio_values = model.generate(descriptions);
   #pieces+=audio_values;

#save audio in file
   name_of_file=str(i);
   name_of_file+='.wav';


   scipy.io.wavfile.write(name_of_file, rate=sampling_rate, data=audio_values[0, 0].cpu().numpy());
   print('iteration=',i);

  729387498723984709827 4
random.py



from random import choice, randint;
import soundfile as sf;
max_len=1250000;

number_of_file=raw_input('number_of_file=');
number_of_file=int(number_of_file);
number_of_iteration=raw_input('number_of_the_iteration=');
number_of_iteration=int(number_of_iteration);
```

```
answer=range(1,number_of_file+1);
sierra=[];
for i in answer:
    string=str();
    string+=str(i)+'.ogg';
    data, number=sf.read(string);
    sierra.append(number);

max_samplerate=max(sierra);
del sierra;

sierra=divmod(len(data),max_len);
integer_part_of_the_division=sierra[0];
remainder_of_the_division=sierra[1];
del sierra;


output=open('output.ogg','w');

for i in xrange(0,number_of_iteration):
    string=choice(answer);
    string=str(string);
    string+='.ogg';
    data, samplerate=sf.read(string);



    if len(data)>max_len:

        sierra=divmod(len(data),max_len);
        integer_part_of_the_division=sierra[0];
        remainder_of_the_division=sierra[1];
        del sierra;

        for i in xrange(0,integer_part_of_the_division):
            number_1=max_len*i;
            number_2=max_len*(i+1);
            data_write=data[number_1:number_2];
            sf.write(output,data_write,max_samplerate);

        number_1=max_len*integer_part_of_the_division;
        number_2=number_1+remainder_of_the_division;
        data_write=data[number_1:number_2];
        sf.write(output,data_write,max_samplerate);

    else:
            sf.write(output, data, max_samplerate);


output.close();
```