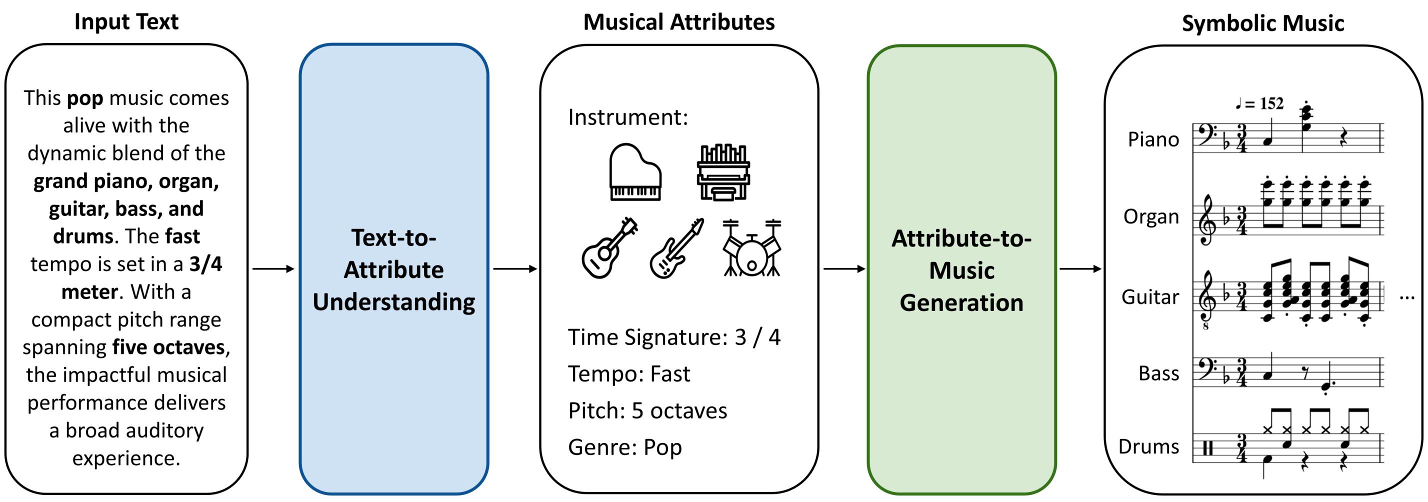
# MuseCoco: Generating Symbolic Music from Text


Input Text
This pop music comes alive with the dynamic blend of the grand piano, organ, guitar, bass, and drums. The fast tempo is set in a 3/4 meter. With a compact pitch range spanning five octaves, the impactful musical performance delivers a broad auditory experience.
Text-to-Attribute Understanding
Musical Attributes
Instrument:
Time Signature: 3 / 4
Tempo: Fast
Pitch: 5 octaves
Genre: Pop
Attribute-to-Music Generation
Symbolic Music
♩= 152
Piano
Organ
Guitar
Bass
Drums
...

## What’s New!

[2023.06.30] **Checkpoints are released!** 📣🎶

[2023.06.01] **We create the repository and release the paper.** 🎉🎵

## Environment

```
# Tested on Linux.
conda create -n MuseCoco python=3.8
conda activate MuseCoco
conda install pytorch=1.11.0 -c pytorch
pip install -r requirements.txt  # g++ should be installed to let this line wor
```

## Attributes

The mapping between keywords used in the code and musical attributes:

```
{
    "I1s2": "Instrument",
    "R1": "Rhythm Danceability",
    "R3": "Rhythm Intensity",
    "S2s1": "Artist",
    "S4": "Genre",
    "B1s1": "Bar",
    "TS1s1": "Time Signature",
    "K1": "Key",
    "T1s1": "Tempo",
    "P4": "Pitch Range",
    "EM1": "Emotion",
    "TM1": "Time"
}
```

## Training

### I. Text-to-Attribute Understanding

#### 1. Construct attribute-text pairs

Switch to the `1-text2attribute_dataprepare` folder

1. Attribute: We provide attributes of the standard test set in text.bin.

2. Construct Text:

   ```
   cd 1-text2attribute_dataprepare
   bash run.sh
   ```

3. Obtain attribute-text pairs (the input dataset for the text-to-attribute understanding model) including `att_key.json` and `test.json` . We have provided the off-the-shelf standard test set in the folder too.

### 2. Train the model

Switch to the `1-text2attribute_model` folder

```
cd 1-text2attribute_model
bash train.sh
```

The checkpoint of the fine-tuned model and `num_labels.json` are obtained.

## II. Attribute-to-Music Generation

### 1. Data processing

Switch to the `2-attribute2music_dataprepare` folder. Then, run the following command to obtain the packed data. Note that `path/to/the/folder/containing/midi/files` is the path where you store the MIDI files, and `path/to/save/the/dataset` is an arbitrary folder you designate to store the extracted data.

```
python extract_data.py path/to/the/folder/containing/midi/files path/to/save/the
```

**Note:** The tool can only automatically extract the objective attributes' values from MIDI files. If you want to insert values for the subjective attributes' values, please input it manually at L40-L42 in `extract_data.py` .

The above commend would tokenize the music and extract the attributes from the MIDI files, and then save the information in 4 files named `Token.bin` , `Token_index.json` , `RID.bin` , `RID_index.json` in your designated folder. Please move those files into `2-attribute2music_model/data` , and switch to `2-attribute2music_model/data_process` , then run the following command to process the data into `train, validation, test` .

```
# The following script splits the midi corpus into "train.txt", "valid.txt" and
```

```
python split_data.py

#The following script binarizes the data in fairseq format.
python util.py
```

### 2. Training

Switch to the `2-attribute2music_model` folder

Run the following command to train a model with approximately 200M parameters.

```
bash train-xl.sh
```

## Inference

## I. Text-to-Attribute Understanding

Switch to `1-text2attribute_model` folder

1. Prepare the text as the format in predict.json.
2. Set `test_file` as the path of `predict.json` in `predict.sh` .
3. Then,

   ```
   bash predict.sh
   ```

   The `predict_attributes.json` and `softmax_probs.json` are obtained.

4. Preprocess the input of the attribute-to-music generation stage for inference After inference, set the path of `predict.json` , `predict_attributes.json` , `softmax_probs.json` and `att_key.json` in `stage2_pre.py` and then,

   ```
   python stage2_pre.py
   ```

   The `infer_test.bin` is obtained as the inference input of the attribute-to-music generation stage.

## II. Attribute-to-Music Generation

Switch to `2-attribute2music_model` folder

5. Download the checkpoint and Prepare it in `checkpoint/linear_mask-1billion`
6. Prepare the input for inference in the folder `data/infer_input/infer_test.bin` from the output of text-to-attribute understanding stage ( `infer_test.bin` ).
7. Run the following command to generate music based on the first 200 samples in `infer_test.bin` .

```
# The following script takes "data/infer_input/infer_test.bin" as input.
bash interactive_1billion.sh 0 200
# bash interactive.sh start_idx end_idx input_name
```

The generated results are located in the folder `generation/`

## III. Evaluate Generated Music

If you'd like to evaluate the generated music, extracted objective attributes can be regarded as gold labels. Switch to the `evaluation` folder and run the following command:

```
# python eval_acc_v3.py --root=2-attribute2music_model/generation/0505/linear_ma
python eval_acc_v3.py --root=PATH_OF_GENERATED_MUSIC
```

The average sample-wise accuracy for objective attributes (ASA) is printed. The accuracy of each objective attribute is in `acc_results.json` . The accurateness of every attribute in each MIDI file is shown in `midiinfo.json` ( `value_dict` refers to the extracted attributes and '0' and '1' in `acc` refer to error and correctness respectively)

## Usage Tips

To maximize your utilization of MuseCoco, here are some valuable tips to enhance your overall experience:

1. You have two options for creating text descriptions: writing them yourself or using the synthesis method with ChatGPT, as mentioned in the paper. We recommend the synthesis method with attribute values. It's easier and aligns better with the data distribution.
2. Please ensure that you utilize the specific attributes mentioned in the paper and implemented in the code when constructing your text descriptions. Using attributes that are not explicitly mentioned may lead to undesired control accuracy. Stick to the specified attributes to achieve the desired level of control.

3. Please use the attribute values we provided, other values or categories for each attribute will bring about undesired control accuracy.
4. Please use the evaluation model above to calculate the control accuracy, and select the samples with the highest accuracy for improved performance. While it is difficult to guarantee 100% control accuracy, it is advisable to automatically filter out some samples to enhance the overall results.

## Citation

```
@article{musecoco2023,
  title={MuseCoco: Generating Symbolic Music from Text},
  author={Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, Jiar
  journal={arXiv preprint arXiv:2306.00110},
  year={2023}
}
```