CHART-BASED ACCOMPANIMENT GENERATION USING PRETRAINED TRANSFORMERS

Maximos Kaliakatsos-Papakostas
Department of Music Technology and Acoustics, Hellenic Mediterranean University
maximoskp@hmu.gr

ABSTRACT

Generating piano accompaniment on a given chart, i.e., melody and chord symbols, is a task that requires knowledge about harmony, rhythm and stylistic nuances. A transformer-based machine learning architecture is explored for tackling this task, where an optional text prompt and the the chords and melody from a chart are passed through respective pretrained transformer encoders and then combined to form an input for the encoder part of an encoder-decoder accompaniment generator. Because of the lack of annotated data, a crude approach to construct chart-like data is followed, where melody and chords in the form of binary pitch class profiles are extracted from any midi file. The results indicate that involving the pretrained components for forming the encoder input improve learning, but the accompaniment results of this approach are problematic, possibly because of the crude data preprocessing and the segregation of the chord-melody parts in the input of the generating tranformer encoder.

1. INTRODUCTION

Aim of this study is to create a system that receives chart information, i.e., chords and a melody, along with an optional text prompt and generate piano accompaniment. Given the scarce availability of chart-format symbolic music data in a wide range of styles, the focus of this work is to crudely create chart-like information from generic MIDI files. The results show that this crude data preprocessing approach, among other factors, might not provide optimal results and a more careful approach is necessary.

2. SYSTEM DESCRIPTION

The examined method comprises the following components: a) a *data preparation* stage, where the melody of a MIDI file is isolated from the remaining ("other") notes; b) an *encoder input preparation* stage, where an optional text prompt, the harmonic information of the "other" and the melody parts are tokenized, fed into the respective pretrained transformer encoders and concatenated form a unified input to the next stage; and c) an encoder-decoder transformer *accompaniment generation* stage, where accompaniment symbolic data are generated based on the unified encoder input produced in the previous step. Figure 1 shows an overview of the system, showing also the pretrained components that remain frozen during training.

Data preparation: MIDI files are considered as inputs, without any assumption about the tracks they include. To construct a chart-like data input format from any midi file, the non percussive tracks are merged into a single, unified pianoroll. If two tracks include notes with the same pitch and onset time, the one with the highest velocity value is retained, regardless of their duration. A simple "skyline" algorithm is applied on the unified pianoroll which results in two distinct chuncks of MIDI information that include the notes of the *melody* and the *other* notes. The Lead-AE [1] could have been used for this task – it will be examined in future iterations of this system.

The melody component is used in its original format, meaning that it is subsequently tokenized in its full MIDI detail in REMI format [2] using MidiTok [3]. The *binary pitch class profile* (bPCP) of the "other" notes is extracted, every quarter note. The bPCP indicates the presence / absence of each pitch class instead of their quantities (as measured in the PCP) within a music segment. In other words, the binary pitch content of the non-melody notes is aggregated every quarter note and the bPCP is retained in a $\{0,1\}^{12\times T}$ matrix, where T is the number of quarter notes within the excerpt at hand.

Encoder input preparation: This stage includes passing all the tokenized data (text, bPCP and melody MIDI) from their respective pretrained transformer encoders. All encoders have been pretrained using the RoBERTa [4] architecture with masked language modeling (MLM). The text [1] does not play any crucial role in this implementation since training is performed on a single dataset with specific characteristics (POP909 [2]). Random pop music-related sentences are used during training and inference.

The bPCP encoder has been pretrained using the GiantMIDI dataset, following a "tiny" architecture with 4 layers each with 4 attention heads and a hidden size of 256, reaching an accuracy of over 95% in MLM. The code for the bPCP is available online [2]. The tokenization process

[1] 'roberta-base' from huggingface.

[2] https://github.com/maximoskp/

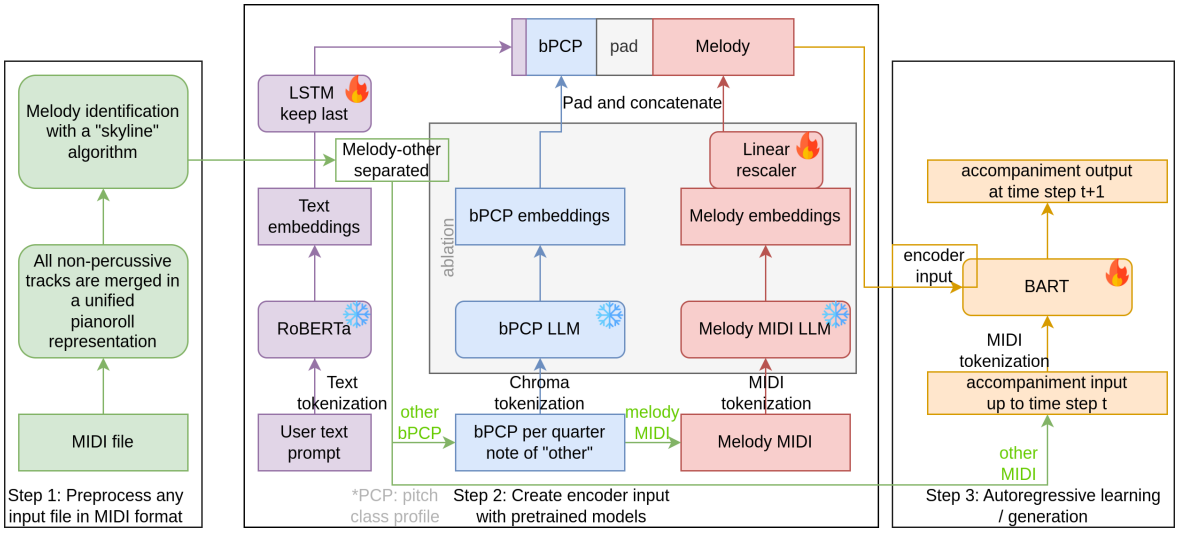

Figure 1. System overview.

is a novel approach that aims to "flatten" each 12D column of the $\{0,1\}^{12\times T}$ rPCP matrix. An obvious choice would be to assign each 12D column a specific token. The apprach followed for bPCP tokenization was to describe C major as `c_seg, c_0, c_4, c_7`, C major without root note `c_seg, c_4, c_7` and C minor `c_seg, c_0, c_3, c_7`, where `c_seg` indicates the beginning of a new "chord" segment, while each `c_i` indicates the existence of the binary pitch class `i` in this segment. This way, we believe, the system would have an easier job identifying the contributing parts of the context of each segment; e.g., a C major segment provides most of the time `c_4`, while C minor `c_3`. The claim that this tokenization process would be superior to a simple name-each-chord approach has not been thoroughly examined and will be included in forthcoming research. The output of the bPCP encoder is an matrix of size $256 \times P$, where P is the number of tokens from the input excerpt.

The melody MIDI encoder [3] follows again a RoBERTa architecture included 8 layers with 8 attention heads each and a hidden size of 512, while it has been pretrained again with the GiantMIDI dataset. A trainable linear layer is employed to map the 512 dimensions to 256, so that all encoder outputs can be concatenated into a single encoder input for the generation stage. The concatenation includes the text single-column final hidden state LSTM output, the bPCP encoder output, a padding with -1 that ensures that the next melody encoder output starts exactly at position 100 – which is enough for the text and bPCP to fit, given the competition bars / time signature conditions.

Accompaniment generation: The concatenated output of the pretrained encoders is provided as input embeddings to a BART [5] encoder-decoder architecture with 8 layers with 16 heads each for both the encoder and the decoder, with a hidden size of 4096. The BART model is trained for autoregressive generation by providing the MIDI format of the "other" part that was generated by the skyline algorithm. The BART system was trained on the POP909 dataset.

[3] `https://github.com/maximoskp/pretrain_MIDI_transformers/tree/main/data_preparation`

3. RESULTS AND DISCUSSION

Model	Epoch 10	20	30	40	50
Embed	.65	.80	.85	.87	.89
Tokens	.47	.59	.62	.63	.64

Table 1. Ablation comparison.

Table 1 shows the training accuracy of BART output when the pretrained encoders were used for preparing the BART encoder input ("Embed" row), versus when they were not used and tokens with padding were used as input directly to the BART encoder ("Tokens" row). It appears that the pretrained models help crucially the convergence of BART. Despite the hight accuracy, however, the generated accompaniment does not seem to always follow the chords on the input chart and is mainly following well-learned harmonic progressions in the dataset that fit the melody notes (sometimes, not even following the melody notes). Among many possibilities for improvement, the next two steps might be: a) employ Lead-AE instead of skyline for making clearer chord-melody separation and b) intertwine sequentially in a timely fashion bPCP tokens with MIDI tokens, using a unified tokenizer to and create a single pretrained model for the two.

4. REFERENCES

[1] Z. Novack, N. Srivatsan, T. Berg-Kirkpatrick, and J. McAuley, "Unsupervised lead sheet generation via semantic compression," *arXiv preprint arXiv:2310.10772*, 2023.

[2] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM international conference on multimedia*, 2020, pp. 1180–1188.

[3] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski, "MidiTok: A python package for MIDI file tokenization," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021. [Online]. Available: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf

[4] Y. Liu, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, vol. 364, 2019.

[5] M. Lewis, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.