MIDITOK: A PYTHON PACKAGE FOR MIDI FILE TOKENIZATION

Nathan Fradet[1,4] **Jean-Pierre Briot**[1] **Fabien Chhel**[2]
Amal El Fallah Seghrouchni[1] **Nicolas Gutowski**[3]
[1] Sorbonne University, CNRS, LIP6, F-75005 Paris, France
[2] ESEO-TECH / ERIS, 49100 Angers, France
[3] University of Angers, LERIA, 49000 Angers, France
[4] Aubay, Boulogne-Billancourt, France
nathan.fradet@lip6.fr

ABSTRACT

This article presents *MidiTok*, a Python package to encode MIDI files into sequences of tokens to be used with sequential Deep Learning models like Transformers or Recurrent Neural Networks. It allows researchers and developers to encode datasets with various strategies built around the idea that they share common parameters. This key idea makes it easy to :1) optimize the size of the vocabulary and the elements it can represent w.r.t. the MIDI specifications; 2) compare tokenization methods to see which performs best in which case; 3) measure the relevance of additional information like chords or tempo changes. Code and documentation of *MidiTok* are on Github [1] .

1. INTRODUCTION

From recent progress in Neural NLP have emerged models and methods, such as Transformer and attention mechanisms, which are also used in most recent music-related models [1]. Using symbolic Music with these models requires, as for text, to *tokenize* this data, i.e. transform it into sequences of tokens to be encoded in embedding vectors afterwards. Lately, research in the field of symbolic music generation highlighted the importance of how music is encoded, i.e. how the notes, its attributes, time and other information are transformed into tokens, grouped and arranged for specific purposes. Most papers legitimately praise the benefits of their encoding strategy compared with others, but sometimes without explicitly stating their key differences or similarities. For this purpose, we created *MidiTok*, an open-source Python package reproducing the most popular tokenization methods with common parameters, allowing the encoding of symbolic music with different strategies, while representing the exact same information with the same accuracy.

[1] github.com/Natooz/MidiTok. MidiTok can be installed with pip.

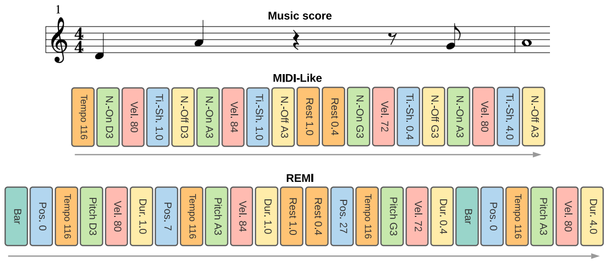

Figure 1. A sheet music and two token representations.

2. TOKENIZATION METHODS

MidiTok features the most known tokenization methods. Thus we introduce them below and point to the related articles for details.

Midi-Like [2], which encodes MIDI messages (Note On, Note Off, Time-shift...) as distinct tokens.

Structured MIDI Encoding introduced with the Piano Inpainting Application [3]. It is similar to Midi-Like but with Duration tokens instead of Note Off, and a recurring pattern of token type successions.

REMI in the Pop Music Transformer [4], represents time with Bar and Position tokens, allowing a model to easily understand recurring music patterns.

Compound Word in CP Transformer [5] is similar to REMI, but uses the associative properties of embeddings to group tokens together. Hence, it reduces the length of the sequence which thus reduces the complexity of the model. This is especially useful for softmax attention mechanism with quadratic complexity.

Octuple [6], similarly to Compound Word, groups tokens together, and includes the current bar and position, tempo and track information to each embedding.

MuMIDI introduced with PopMAG [7], groups tokens of each note's attributes and includes a beat and bar wise positional encoding.

The motivation of our work is twofold: 1) Take advantage of each best tokenization method, 2) Benefit from a fair comparison framework (see Section 3).

A.t.o.w. MidiTok only considers 4/4 time signature, i.e. bars are always considered with 4 beats.

3. COMMON TOKENIZATION PARAMETERS

In *MidiTok*, each tokenization method contains four parameters ensuring that notes and their attributes are represented with the same precision. Figure 2 shows the common operation flow of *MidiTok*. The following flexible parameters allow to reduce the length of a vocabulary $\mathcal{V}$, which is often desired to reduce the perplexity of a model's predictions:

- `pitch_range`: Specifies the minimum and maximum pitch values. Whereas pitch can take values from 0 to 127, the General MIDI 2 specifications [2] details the recommended pitch ranges for each program (i.e., instrument). In *MidiTok* the default pitch range starts from 21 to 108, which covers every recommended pitch values from the specifications.

- `nb_velocities`: Is the number n of velocity values to represent. The velocities of a MIDI, from 0 to 127, will be quantized into n values.

- `beat_res`: Specifies the sample rates per beat and the beat ranges to apply them. This parameter is given as a map, for instance `{(0,4): 8, (4,12): 4 }` where beats 1 to 4 will be quantized at 8 samples per beat, and beats 5 to 12 at 4 samples per beat. This allows the creation of accurate duration and time-shift values for short notes and less accurate values for longer notes. The latter tend to occur less often in most cases.

- `additional_tokens`: Represents the additional tokens to be included, they are described in the next Section.

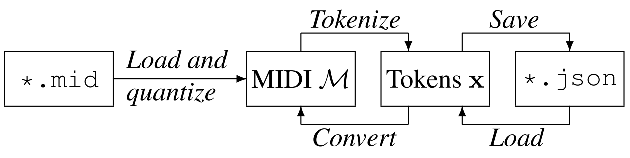

Figure 2. The main operation flow of MidiTok.

4. ADDITIONAL TOKENS

Similarly to the featured encoding methods, *MidiTok* proposes to include additional tokens within the sequences. We depict them below:

- `chord`: Indicates that a chord is being played at the current time step. In *MidiTok* we designed a simple and efficient time-step based algorithm to detect chords.

- `rest`: Acts as a "Time-Shift" event moving the time from section where no note is being played. Rests are key elements in music as they are part of the overall texture. Similarly to sheet music where rests help a musician to read, it might be beneficial to explicitly inform a deep learning model where the rests are within a music part, instead of just moving in time.

- `tempo`: Informs of the current tempo. This additional token type operates with two parameters: the tempo range (minimum and maximum tempo) and the number of tempo values to quantize this range.

Furthermore, MidiTok offers to include `program` tokens in the vocabulary if ones need them, for multitrack tasks for instance. We plan to include Control Change messages to bring further more expressive information and allow a model to generate MIDI effects such as the sustain pedal or modulation wheel.

5. CASE STUDY AND FUTURE WORK

This section presents a short case study of *MidiTok*. When using transformer networks, sequence length is a well known bottleneck due to the quadratic complexity of the dot product attention mechanism. With this concern in mind, researchers found ways to reduce sequence lengths, creating methods like *Compound Word* [5] or *Octuple* [6].

We tokenized the MetaMIDI [3] [8] and GiantMIDI [9] datasets with four methods and measured the average sequence lengths, reported in Table 1. We used pitches from 21 to 108, 32 velocities, beat resolutions of `{(0,8): 8, (8,16): 4, (16,32): 2}` and no additional tokens.

We first notice that the CP Word strategy reduces the length by more than half in every case. Other strategies give similar results, which depends on the characteristics of the MIDI files. MIDI-Like will be shorter than Structured for MIDIs with chords, and vice versa.

These numbers are however to be considered depending on other criteria. We begin to see efficient transformers [10, 11] taking benefit of the kernel trick to reduce the complexity of attention from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$ with very few assumptions. These models are used in [3, 5] and show good results that could question the interest of sequence length reduction strategies. With these questions, our current research aims to measure which tokenization methods perform best for various cases, tasks, attention mechanisms, positional encodings and datasets.

Table 1. Average length of token sequences.

Dataset	M-Like	Structured	REMI	CP Word
MetaMIDI	1772	1791	1701	806
GiantMIDI	13281	14269	12761	5626

6. CONCLUSION

This article presents *MidiTok*, a convenient package for tokenizing MIDI files featuring multiple strategies with common and flexible parameters. We expect it to facilitate the task of tokenizing MIDI files for research topics such as symbolic music generation or information retrieval.

[2] See the MIDI Manufacturers Association website

[3] Actually a subset of 33,027 distinct MIDI files from a weighted graph matching of the MIDI-Audio scores.

7. ACKNOWLEDGEMENT

We thank Eric Remilleret and Didier Giot for their support, and Aubay for funding this work.

8. REFERENCES

[1] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep Learning Techniques for Music Generation*, ser. Computational Synthesis and Creative Systems. Springer International Publishing, 2020. [Online]. Available: https://www.springer.com/gp/book/9783319701622

[2] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, p. 955967, 2018.

[3] G. Hadjeres and L. Crestel, "The piano inpainting application," in *arXiv: 2107.05944*, 2021.

[4] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 11801188. [Online]. Available: https://doi.org/10.1145/3394171.3413671

[5] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs." AAAI Conference on Artificial Intelligence, 2021.

[6] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," in *arXiv: 2106.05630*, 2021.

[7] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 11981206. [Online]. Available: https://doi.org/10.1145/3394171.3413721

[8] J. Ens and P. Pasquier, "Building the metamidi dataset: Linking symbolic and audio musical data," in *Proceedings of 22st International Conference on Music Information Retrieval, ISMIR*, 2021.

[9] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," in *Transactions of the International Society for Music Information Retrieval, ISMIR*, 2021.

[10] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Ua6zuk0WRH

[11] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.