N-Gram Unsupervised Compoundation and Feature Injection for Better Symbolic Music Understanding

Jinhao Tian[1,2,†], **Zuchao Li**[1,2,*], **Jiajia Li**[3,4,†], **and Ping Wang**[3,4,*]
[1]**National Engineering Research Center for Multimedia Software,**
School of Computer Science, Wuhan University, Wuhan, 430072, P. R. China
[2]**Hubei Luojia Laboratory, Wuhan 430072, P. R. China**
[3]**Center for the Studies of Information Resources, Wuhan University, Wuhan 430072, China**
[4]**School of Information Management, Wuhan University, Wuhan 430072, China**
{jinhaotian,cantata,zcli-charlie,wangping}@whu.edu.cn

Abstract

The first step to apply deep learning techniques for symbolic music understanding is to transform musical pieces (mainly in MIDI format) into sequences of predefined tokens like note pitch, note velocity, and chords. Subsequently, the sequences are fed into a neural sequence model to accomplish specific tasks. Music sequences exhibit strong correlations between adjacent elements, making them prime candidates for N-gram techniques from Natural Language Processing (NLP). Consider classical piano music: specific melodies might recur throughout a piece, with subtle variations each time. In this paper, we propose a novel method, NG-Midiformer, for understanding symbolic music sequences that leverages the N-gram approach. Our method involves first processing music pieces into word-like sequences with our proposed unsupervised compoundation, followed by using our N-gram Transformer encoder, which can effectively incorporate N-gram information to enhance the primary encoder part for better understanding of music sequences. The pre-training process on large-scale music datasets enables the model to thoroughly learn the N-gram information contained within music sequences, and subsequently apply this information for making inferences during the fine-tuning stage. Experiment on various datasets demonstrate the effectiveness of our method and achieved state-of-the-art performance on a series of music understanding downstream tasks. The code and model weights will be released at https://github.com/WouuYoauin/NG-Midiformer.

* Corresponding author. † Equal contribution. This work was supported by the National Natural Science Foundation of China (No. 62306216), the Natural Science Foundation of Hubei Province of China (No. 2023AFB816), the Fundamental Research Funds for the Central Universities (No. 2042023kf0133), National Natural Science Foundation of China [No. 72074171] [No. 72374161].

Introduction

Symbolic Music Understanding, distinct from audio-based understanding (Nam et al. 2018), involves the computational analysis and interpretation of symbolic music sequences. This understanding aids tasks like music generation (Briot, HADJERES, and Pachet 2019) and Music Information Retrieval (MIR)(Casey et al. 2008). The process starts by converting music pieces into sequences of predefined tokens, representing musical events such as note pitch and tempo. These sequences are then processed using neural networks, notably the Transformer(Vaswani et al. 2017). Music tokenization methods fall into two categories: direct conversion methods like MIDI-LIKE (Oore et al. 2020) and REMI (Huang and Yang 2020), and methods employing expansion and compression techniques (Li et al. 2021), such as Compound Word (CP)(Hsiao et al. 2021) and OctupleMIDI(Zeng et al. 2021).

Direct conversion methods like MIDI-LIKE and REMI maintain music event atomicity but tend to produce extended sequences, weakening token dependencies. On the other hand, techniques like CP and OctupleMIDI, which utilize expansion and compression, yield shorter sequences rich in contextual information. However, they risk missing critical short-range dependencies between adjacent elements and often introduce redundant "[PAD]" tokens or replicate neighboring musical tokens, diminishing the sequence's informational value. It's pivotal to note that music events often exhibit co-occurrence regularities, with certain events frequently appearing together, mirroring the music's semantic depth. This regularity necessitates a tokenization method that resonates with the unique characteristics of music events. Interestingly, music event atoms exhibit combinatory traits akin to characters in natural language. Motivated by this parallel, we propose an unsupervised compoundation method based on frequency. By analyzing the co-occurrence patterns of music events, we craft sequences using a frequency-driven vocabulary, ensuring a more con-

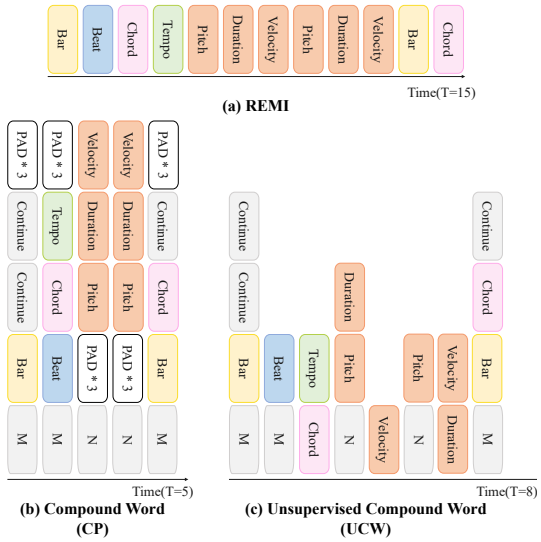

Figure 1: An example illustrating the relationships and distinctions among REMI, CP, and UCW. Here, M denotes the family "metric", and N denotes the family "note".

cise sequence length than REMI while preserving the inherent correlations of compound words.

The success of sequence neural networks in NLP across diverse languages has provided valuable insights for symbolic music understanding (Vaswani et al. 2017; Devlin et al. 2019). Transferring these models to symbolic music has proven effective in music generation and understanding (Hsiao et al. 2021; Chou et al. 2021; Zeng et al. 2021). However, existing models seldom undertake further processing to handle the strong correlation between consecutive adjacent musical elements, a feature that distinguishes music from language. Some methods even inadvertently reduce this correlation. Music often showcases repeating pattern both locally and across sections. The N-gram model, designed to capture local patterns, aligns well with this correlated nature of music and can be instrumental in identifying and leveraging these repetitions for enhanced the representation of symbolic music. Furthermore, music's hierarchical structure, ranging from individual notes to entire compositions, can be captured by N-grams, especially at smaller scales. Despite this alignment, few current models utilize N-grams in symbolic music understanding. We introduce an N-Gram Transformer encoding that capitalizes on the strong correlation of neighboring music events, improving music sequence understanding.

Unsupervised compoundation, a novel music tokenization technique, extracts N-gram information between neighboring music events. It synergizes ideas from previously mentioned tokenization methods. Initially, continuous music elements from the same "family" are grouped into a "word", as defined in Compound Word (Hsiao et al. 2021). Here, a "family" refers to a specific category of musical event, like pitch, duration, and velocity of a note, all under the "note" family. Other families include metric and track, each representing distinct musical categories. Subsequently, Byte Pair Encoding (BPE), an unsupervised method, segments these "words". In the REMI sequence context, this means frequently occurring neighboring music events from the same family merge into a token, with each event akin to a "character" in a "word". Notably, our method omits the [PAD] token present in CP. Figure 1 delineates the relationships among REMI, CP, and our Unsupervised Compound Word (UCW). This technique not only shortens sequences compared to REMI but also sidesteps the excessive, often meaningless, [PAD] tokens seen in CP.

After UCW sequence construction, an encoder is employed to extract N-gram features, enhancing the model's symbolic music understanding via N-gram tokens. Our model, NG-Midiformer, incorporates a primary encoder for the input music sequence and an N-gram Transformer encoder to harness N-gram sequence information. This encoder, rooted in the Transformer architecture, is tailored to exploit N-gram information. To operationalize this, we commence by deriving N-grams from the given corpus and obtaining their frequencies, thereby constructing an N-gram vocabulary. For every input music sequence, we extract the relevant N-grams and pinpoint their positions within this established vocabulary. This comprehensive approach aligns with the intricate nature of symbolic music, and can capitalizes on the global structure inherent in musical compositions.

Our method's distinctiveness emerges from its integration mechanism. At every primary encoder layer, the output from the N-gram encoder is integrated, enriching the representation with N-gram contextual information. Specifically, for each N-gram, its hidden layer output is multiplied by its frequency, normalized, and then added to the primary encoder's hidden layer output, ensuring a comprehensive musical context understanding.

We pre-train NG-Midiformer on a large-scale, unlabeled symbolic music dataset, subsequently assessing its efficacy across six downstream tasks: composer, emotion, genre, and dance classification, as well as velocity prediction and melody classification. Our NG-Midiformer outperformed prior state-of-the-art models across these tasks, showcasing its superiority on datasets like Pianist8 (joann8512 2021), EMOPIA (Hung et al. 2021), GTZAN (Sturm 2013), Nottingham (Allwright 2003), and POP909 (Wang* et al. 2020). This establishes our method as a robust new benchmark in symbolic music understanding.

Related Work

Music Tokenization

Musical compositions, akin to natural language, have "grammatical" and "semantic" structures, enabling their representation as structured sequences (Patel 2003; Li and Ogihara 2006). While the pianoroll method encodes music into matrices differentiating pitch and time, it often requires fixed-length music event processing, reducing efficiency. Currently, MIDI files have emerged as a dominant symbolic representation, being lightweight and capturing essential musical elements.

Methods like MIDI-LIKE (Oore et al. 2020) and REMI (Huang and Yang 2020) provide detailed musical information but result in extended sequences. Compound Word (CP) (Hsiao et al. 2021), a refinement of REMI, groups related music elements, shortening sequences. OctupleMIDI (Zeng et al. 2021), an evolution of REMI and CP, offers structured encoding for diverse music types, being more concise than CP. However, both CP and OctupleMIDI introduce substantial filler information in tokens and employ multiple independent embedding layers, potentially weakening the relationship modeling between music events.

To tackle these challenges, we present an unsupervised compoundation method. This new tokenization strategy consolidates frequent adjacent music elements into one token, streamlining embedding processes and boosting efficiency. Using N-Gram data from music event groups, our approach delves deeper into music's semantic layers.

Symbolic Music Understanding

The development of symbolic music understanding is closely related to advancements in NLP techniques (Jackendoff 2009). This is because both music and natural language can be represented as sets of symbols with certain structures and rules. Based on Word2Vec (Mikolov et al. 2013a,b) in NLP, researchers have grouped different musical notes together and treated them as a single unit or "word". They then trained deep learning models on the resulting sequences (Hirai and Sawada 2019).

The emergence of models like Transformers (Vaswani et al. 2017), BERT (Devlin et al. 2019), GPT-3 (Floridi and Chiriatti 2020) and so on (Zhang et al. 2020) have not only revolutionized NLP but also enriched symbolic music understanding. This is evident in the enhancements seen in models like Transformers-XL (Dai et al. 2019) and CP Transformers (Hsiao et al. 2021). MIDI-Bert (Chou et al. 2021) and MusicBert (Zeng et al. 2021), both large-scale music pretraining models, exemplify the successful adaptation of NLP techniques, particularly BERT's architecture and Roberta's structure (Liu et al. 2019), to symbolic music. These models simply transfer the methods of NLP to symbolic music sequences and make some modifications.

However, music's inherent co-occurrence patterns, such as consistent events from certain chords, necessitate a tailored approach. Meanwhile, N-grams possess the capacity to capture such co-occurrence patterns (Brown et al. 1992; Sari, Vlachos, and Stevenson 2017; Shafiq, Khayam, and Farooq 2008). To this end, we champion an encoding structure accentuating N-Gram features to enhance the representation of symbolic music. While NLP's progress is commendable, direct N-Gram technique adoption remains sparse. Inspired by ZEN (Diao et al. 2020), our proposed N-Gram Transformer structure discerns N-Gram relationships in symbolic music sequences, bridging this gap.

NG-Midiformer

Symbolic music events have highly pronounced local dependencies, and N-grams are particularly suitable for sequences with strong local dependencies. In this section, we propose a NG-Midiformer model, a powerful architecture for processing symbolic music sequences. The core concept underlying our model is N-gram, which consists of two key aspects: transforming music into appropriate tokens according to N-gram within the events family, and using N-gram between tokens within the sequence to enhance the understanding ability of the model.

Music Tokenization

To achieve symbolic music understanding using deep learning models like the Transformer, musical pieces must first be converted into symbolic element sequences. Given a musical piece X in MIDI format, a mapping function f transforms it into a sequence S using predefined music tokens e from vocabulary V^{raw}:

$$S = f(X) = \{e_1, e_2, ..., e_N\} \tag{1}$$

where N represents the sequence length, and e_i is the i-th predefined musical token.

However, the self-attention mechanism in the Transformer struggles with longer sequences, such as those produced by the REMI method. To mitigate this, the CP method aggregates music events from the same family into compound words.

Given a REMI sequence $\mathcal{R}$, the CP representation is constructed as follows. Each CP_j is represented as:

$$CP_j = [e^{cp}_{j,1}, e^{cp}_{j,2}, \ldots, e^{cp}_{j,K}] \tag{2}$$

where K represents a fixed number of music element types in a compound word.

As for the definition of "family", We partition these K types into several non intersecting families with each representing a specific music family, such as note and metric. For instance, if $\mathcal{K}$ represents the type set corresponding to K and is partitioned into c families, then the relationship among these families adheres to the following:

$$\begin{gathered} \forall i \neq j, \mathcal{K}_i \cap \mathcal{K}_j = \emptyset \\ \bigcup_{i=1}^{c} \mathcal{K}_i = \mathcal{K} \end{gathered} \tag{3}$$

where $\mathcal{K}_i$ is the i-th family.

For each CP_j, identify a continuous segment in $\mathcal{R}$ where all elements belong to the same family. Let's denote this segment as $\mathcal{S}_j$. Then the elements $e^{cp}_{j,t}$ in CP_j are then defined by:

$$e^{cp}_{j,t} = \begin{cases} \mathcal{R}_{\mathcal{F}(j,t)}, & \text{if } \begin{matrix} t \text{ matches the family of } \mathcal{S}j \\ \text{and } \mathcal{R}_{\mathcal{F}(j,t)} \text{ is in } \mathcal{S}_j \end{matrix} \\ \text{[PAD]}, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{F}(j,t)$ is a mapping function that converts the token index j and inner token index t in a CP into the index in the original REMI sequence.

This method may lose crucial short-range dependency information between neighbouring elements as it might not capture the full relationships within a CP token.

To enhance existing music tokenization methods, we introduce an unsupervised compoundation approach tailored for music event representation. This method employs variable-length token design, grouping frequent music events within families by co-occurrence frequency. Unlike CP, we utilize a unified embedding based on segmented compound subwords, offering a closer semantic representation of music. To construct the sequence, we adopt the concept of creating sub-word units from the domain of NLP. Several methods are available for creating sub-word units in NLP, such as SentencePiece (Kudo and Richardson 2018), WordPiece (Wu et al. 2016), Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016) and so on (Zhang et al. 2019). The first two methods are similar to BPE with minor variations in implementation techniques. Therefore, we leverage the unsupervised BPE method to construct our UCW sequences. Combining the ideas of REMI and CP, we construct music elements in the same way as REMI. Then, neighbouring elements belonging to the same "family" in REMI are merged together as a family token according to the concept of "family" in CP. Since we only compound the real music events according to co-occurrence frequency, [PAD] token presented in CP is not needed. Prior to constructing UCW, it is necessary to get the corresponding REMI vocabulary $\mathcal{V}_{REMI}$ and set the size of the UCW vocabulary V_{UCW} artificially. This way, the UCW token can be represented as:

$$\begin{aligned} &UCW_j = e^{ucw}_{j,1} \odot e^{ucw}_{j,2} \odot ... \odot e^{ucw}_{j,k}, \\ &\forall t \in [1,k], t \text{ corresponds to the family of } \mathcal{S}j \\ &e^{ucw}_{j,1} \odot ... \odot e^{ucw}_{j,k} \in V_{UCW} \\ &e^{ucw}_{j,1} \odot ... \odot e^{ucw}_{j,k+1} \notin V_{UCW} \end{aligned} \tag{5}$$

where $\odot$ represents the actual symbol merging operation, which combines two symbols into one symbol and treats it as a single token in subsequent inputs, rather than representing multiple tokens that are concatenated together like in CP. $freq(\cdot)$ represents a frequency counting function, k is a variable length for a UCW.

Using Algorithm 1, we derive the UCW vocabulary from our music event corpus. A comprehensive construction example can be found in Appendix A. Through this, we've effectively reduced the sequence length than REMI while retaining the musical structure, demonstrating the power of unsupervised compoundation in symbolic music representation. As noted by (Hsiao et al. 2021), CP sequences are typically 30%-50% the length of REMI sequences. With UCW, sequence length hinges on the BPE vocabulary size; for instance, a size of 1000 results in sequences 165% the length of CP and 70%-80% of REMI. It is worth noting that the V_N here is artificially set. Importantly, UCW, by effectively grouping neighboring elements, captures robust correlations, yielding semantically richer tokens that enhance dependency detection in our model. Similar to the BPE algorithm in NLP, it strikes a balance between REMI and CP.

Algorithm 1: Unsupervised Compoundation Construct Algorithm

Require: Music event sequence corpus $\mathcal{C}$, and the UCW's vocab size V_{UCW}.

$\mathcal{C}_{UCW} \leftarrow$ Group the events into families from $\mathcal{C}$
$\mathcal{V}_{UCW} \leftarrow \mathcal{V}_{REMI}$
while $\mathcal{V}_{UCW}.length < V_{UCW}$ **do**
  for UCW in $\mathcal{C}_{UCW}$ **do**
    for $i \leftarrow 1$ to $UCW.length - 1$ **do**
      $span \leftarrow (UCW[i], UCW[i+1])$
      $freq[span] \leftarrow freq[span] + 1$
    end for
  end for
  $p \leftarrow$ the event pair with highest frequency in $freq$
  $\mathcal{V}_{UCW} \leftarrow \mathcal{V}_{UCW} \cup p$
  $\mathcal{C}_{UCW} \leftarrow$ replace all p in $\mathcal{C}_{UCW}$ with a new symbol
end while
for each sequence s in $\mathcal{C}$ **do**
  Initialize an empty Set $\mathcal{S}$
  for $i = 1$ to $s.length$ **do**
    if $(s[i], s[i+1])$ not in $\mathcal{S}$ **then**
      Add the pair $(s[i], s[i+1])$ in $\mathcal{S}$
    end if
  end for
  while $\mathcal{S}$ is not empty **do**
    Remove the pair $(s[m], s[m+1])$ from $\mathcal{S}$
    if the pair $(s[m], s[m+1])$ appears in $\mathcal{V}_{UCW}$ **then**
      Replace all $(s[m], s[m+1])$ with the new symbol p
      add $(s[m-1], s[n])$ and $(s[n], s[m+2])$ in $\mathcal{S}$
    end if
  end while
  Append the segmented sequence s to the segmented corpus $\mathcal{C}'$
end for

Ensure: Corpus of UCW $\mathcal{C}'$, UCW's vocabulary $\mathcal{V}_{UCW}$

N-gram Transformer Encoder

After constructing the UCW input sequence, the next step is to utilize an encoder for feature extraction that can effectively analyze and comprehend the music. In this section, we propose the use of an N-gram Transformer encoder, which is first pre-trained on a large-scale unlabeled UCW sequence for self-supervised learning, and then fine-tuned for downstream tasks related to symbolic music understanding.

Our implementation of the N-Gram Transformer encoder is based on CP Transformer, which is a recent development in this field. However, using each CP as a single token can lead to a large vocabulary size, which may negatively impact the model's performance. To avoid this issue, CP transformers have adopted an expansion-compression approach presented in (Rae et al. 2019). This involves treating every element $e^{cp}_{j,k}$ in a CP as a separate token and embedding concatenation to integrate music elements.

In contrast, our approach views each UCW token as a locally semantically complete unit. We then transform these UCW tokens into input embeddings:

$$\begin{aligned} X_j &= \textsc{Emb}(UCW_j) \\ \overrightarrow{X_j} &= X_j + \textsc{PosEmb}(j) \end{aligned} \tag{6}$$

where EMB converting each UCW token into a dense vector

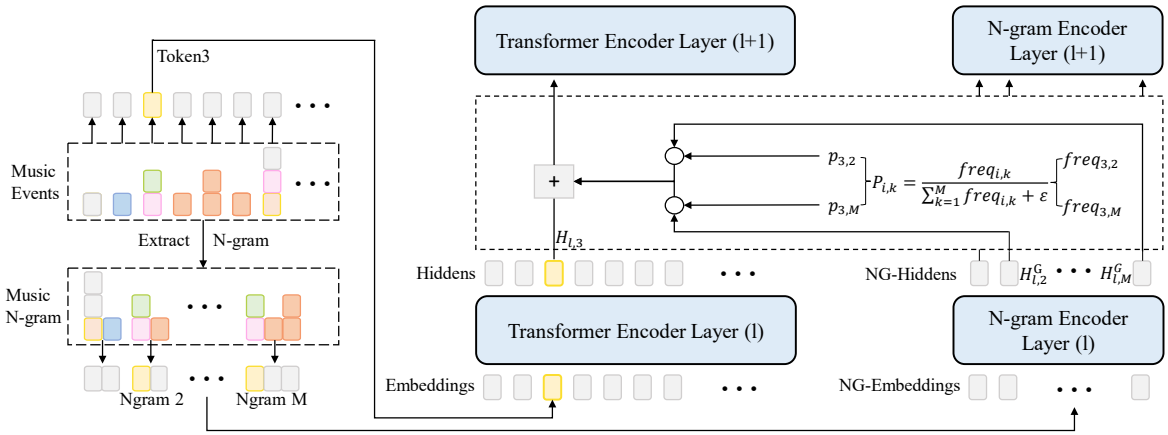

Figure 2: The overall architecture of our proposed N-gram Transformer Encoder. It only displays part of the layers of the model. In reality, after multiple layers of computation, the left Transformer Encoder will access different layer structures to accomplish specific pre-training or downstream tasks, such as Masked Language Modeling (MLM), and Sequence Classification.

representation; PosEmb utilize relative position encoding, assigns vectors based on the relative distances between tokens in the sequence. $\overrightarrow{X_j}$ represents the final input of the j-th UCW into the Transformer layers.

Our N-gram Transformer encoder, visualized in Figure 2, augments attentional encoding by harnessing N-gram relationships between tokens. The standard Transformer encoding is formulated as:

$$H_l = \text{FFN}(Softmax(\frac{QK}{\sqrt{d_k}})V + H_{l-1}), \tag{7}$$

where H_{l-1} and H_l represent the output of $l-1$-th and l-th layer, respectively. $Q = W^Q H_{l-1}$, $K = W^K H_{l-1}$, $V = W^V H_{l-1}$, and $H_0 = \overrightarrow{X}$. d_k is the dimension of the key vectors, usually set as the hidden layer dimension divided by the number of attention heads.

To harness N-gram information in both pre-training and fine-tuning, we extract N-grams from the UCW sequence corpus, forming an N-gram vocabulary, denoted as V_N. Initially, we obtain the frequency of each N-gram occurrence in the corpus and remove low-frequency N-grams to reduce the size of V_N. For each input UCW sequence, we extract all N-grams and select a subset of them contained within the input. We then match these extracted N-grams with the previously obtained N-gram vocabulary V_N, forming a sequence $G = \{g_1, g_2, ..., g_M\}$, where M represents the maximum number of N-grams that can be included in a single input sequence. These N-grams are ranked by their frequency in the corpus, ensuring top N-grams in the input are more frequent. Similar to encoding sequences of music elements, input sequences of N-grams also need to be converted into embeddings for the encoder. We encode each N-gram token similarly as in the token Transformers encoder. Specifically,

$$\begin{aligned} X_j^G &= \text{Emb}^{NG}(g_j), \overrightarrow{X_j^G} = X_j^G + \text{PosEmb}(j) \\ H_l^G &= \text{FFN}(Softmax(\frac{Q^G K^G}{\sqrt{d_k}})V^G + H_{l-1}^G), \end{aligned} \tag{8}$$

where $Q^G = W^Q H_{l-1}^G$, $K^G = W^K H_{l-1}^G$, $V^G = W^V H_{l-1}^G$, and $H_0^G = \overrightarrow{X^G}$; Emb^{NG} converting each N-gram token into a dense vector representation;

Our current approach encodes N-Gram token sequences, aiming to integrate N-Gram information into music sequence encoding. We introduce the N-gram Position Matrix Injection (NPMI) method. With NPMI, we create a matrix called N-gram position matrix (P), which is an $M \times N$ matrix that records the positions and frequencies of each N-gram extracted from the input sequence. This matrix is constructed using the previously extracted N-gram vocabulary and the corresponding N-grams from each input sequence. The N-gram position matrix captures the alignment between N-Grams and their original music sequence, which forms the core of the N-gram Transformer encoder that enhances the model's understanding capabilities using N-gram information. Specifically, based on the position of each N-gram in the sequence, we assign values to the matrix as:

$$P_{i,j} = \begin{cases} freq(g_j), & UCW_i \in g_j \\ 0, & UCW_i \notin g_j \end{cases} \tag{9}$$

where UCW_i denotes the i-th music token, g_j denotes the j-th n-gram token, and $freq(g_j)$ denotes the frequency of g_j in V_N.

To determine the relative frequencies of N-grams, we normalize by dividing each N-gram's frequency by the total frequency of all N-grams at the same position. This normalization captures the N-gram distribution:

$$P_{i,j} = \frac{P_{i,j}}{\sum_{k=0}^{M} P_{i,k} + \varepsilon} \tag{10}$$

where ε is a small constant (1×10^{-10}) to prevent division by zero.

After converting the tokens of both the music input sequences and the N-gram sequences into embeddings and constructing the N-gram position matrix P, we combine the representations of the music tokens and their corresponding N-grams. For each layer in the Transformer, the music token representation H_l is updated by adding the corresponding N-gram representations H_l^G :

$$H_{l,i} = H_{l,i} + \sum_{t \in A} H_{l,t}^G \times P_{i,t} \tag{11}$$

where A is the set of indexes of N-gram tokens that correspond to the music token UCW_i, $H_{l,t}^G$ indicate the representation of t-th N-Gram in l-th layer. Notably, our N-Gram encoding has fewer layers than the music token encoding. Thus, we only add representations for matching layers and skip addition for higher layers.

In fact, for the sake of computational efficiency, we have implemented this injection as:

$$H_l = H_l + P \times H_l^G \tag{12}$$

Experiments

Setup and Downstream Tasks

Our model undergoes a two-stage training: pre-training on a symbolic music dataset, and fine-tuning for six specific downstream tasks. Details on the dataset can be found in Appendix . We set the UCW vocabulary size at 1000, transforming MIDI music files into UCW sequences, and the resultant sequence length is just 165% of the CP sequence. For our N-gram approach, we extracted an N-gram vocabulary V_N from the pre-trained corpus. Each N-gram was indexed and its frequency recorded. We chose an N-value of 4, excluding N-grams with frequencies below 200.

We used the same hyper-parameters as the MIDI-Bert model (Chou et al. 2021), which has a 12-layer structure with 12 self-attention heads, and a hidden layer size of 768 for each self-attention layer. We set a sequence length of 512 for both training stages. The N-gram encoder in our model has a 6-layer structure, with each UC sequence corresponding to an N-gram sequence of length 128. Pre-training took 44 hours (about 128k steps) on 4 NVIDIA GeForce RTX 3060 GPUs, using the AdamW optimizer and a learning rate that warmed up over the initial 1k steps. Our pre-training employed Masked Language Modeling (MLM), masking 15% of input tokens for prediction. Notably, MIDI-Bert (Chou et al. 2021) and CP Transformer (Hsiao et al. 2021) differ in constructing CP sequences. While MIDI-Bert uses pitch, duration, sub-beats, and bars (CP-4), CP Transformer incorporates all seven musical elements (pitch, duration, velocity, bar, beat ,chord and tempo) with padding (CP-7). For a fair comparison, we experimented with both settings, creating UCWs (UCW-4, UCW-7) corresponding to CP-4 and CP-7 based on input music events.

We assessed our model across two main categories of tasks. For sequence classification, we focused on Composer (Com), Emotion (Emo), Genre (Gen), and Dance (Dan) classification. Meanwhile, Melody extraction (Mel) and Velocity classification (Vel) were our primary token classification tasks. A detailed introduction for these six downstream tasks and their corresponding datasets can be found in Appendix .

For all tasks, we fine-tuned our pre-trained model for up to 15 epochs, maintaining a consistent sequence length of 512 CP tokens. Given that a CP might encompass multiple UCWs, we input only the initial 512 UCWs during fine-tuning, which might limit the musical input in our approach. For token-level tasks, we segmented the sequence into 512 UCWs for each token's classification. However, this UCW sequence retains approximately 60% of the CP data, potentially putting our model at a disadvantage.

Throughout both pre-training and fine-tuning, we designated 90% of each task's dataset for training and the remaining 10% for validation.

Main Analysis

Table 1 contrasts the NG-Midiformer's performance with the baseline MIDI-Bert model across all tasks. We aligned our comparison with the CP-4 and CP-7 datasets, leading to our UCW sequences UCW-4 and UCW-7. While CP-4 captures basic musical elements, CP-7 provides a comprehensive view of piano track data. However, CP-7 and UCW-7 aren't apt for velocity and melody classification due to potential data leakage.

For sequence-level classification tasks, we standardized input music sequences to match the length of 512 CP tokens to ensure a fair comparison. In our method, only the initial 60% of the sequence is utilized during inference.

Yet, our NG-Midiformer, leveraging unsupervised compoundation and N-gram Transformer encoding, consistently outperformed both MIDI-Bert and RNN models in nearly all tasks. Specifically, we observed performance boosts of +8.22%, +5.67%, +15.08%, +6.17%, and +0.4%, setting new state-of-the-art results in five tasks, including composer, emotion, genre, dance classification, and velocity prediction.

These outcomes underscore the value of N-gram information in enhancing symbolic music understanding. The fusion of UCW and N-Gram Transformer encoding surpasses the CP or REMI tokenization methods used by MIDI-Bert or RNN (Chou et al. 2021). Notably, UCW-7 demonstrated superior performance over UCW-4, indicating that richer information aids the model's musical comprehension.

While our model excelled in many areas, it fell short in Melody classification. This limitation might stem from using only about 60% of the available information in UCW sequences compared to CP. This might exclude crucial melodic information present in the latter part of sequences, which can be especially important for melody classification. Additionally, our labeling method assigns identical labels to multiple UCWs within the same CP group. This means the model faces increased inference demands in token-level classification tasks, potentially affecting melody classification performance.

Table 1: The testing accuracy (in %) of various tokenization methods and models on 6 different downstream tasks. Symbol (*) indicates that the corresponding experimental results were replicated by us.

Model	Token	Sequence Classification				Token Classification	
		Composer	Emotion	Genre	Dance	Velocity	Melody
MidiGPT (Ferreira, Lelis, and Whitehead 2020a)	CP	-	61.88	-	-	-	-
RNN (Chou et al. 2021)	REMI	51.97	53.46	-	-	44.56	89.96
	CP-4	60.32	54.13	-	-	43.77	88.66
OM-MIDI-Bert (Liu, Xu, and Xu 2022)	REMI	82.41	75.58	-	-	50.82	92.01
	CP-4	75.40	68.81	-	-	53.42	**97.87**
RoAR (Li et al. 2023)	REMI	78.48	73.73	-	-	51.40	91.84
	CP-4	80.95	76.15	-	-	53.73	97.59
MIDI-Bert (Chou et al. 2021)	REMI	67.19	67.74	-	-	49.02	90.97
	CP-4	78.57	67.89	50.49(*)	57.02(*)	51.63	96.37
	CP-7	84.16(*)	72.72(*)	48.65(*)	46.89(*)	-	-
NG-Midiformer	UCW-4	90.63	73.44	56.67	55.05	**54.13**	92.31
	UCW-7	**90.63**	**81.25**	**65.87**	**63.19**	-	-

Table 2: The testing accuracy (in %) of various tokenization methods on 6 different downstream tasks with MIDI-Bert.

Model	Token	Sequence Classification				Token Classification	
		Com	Emo	Gen	Dan	Vel	Mel
MIDI-Bert	REMI	67.19	67.74	-	-	49.02	90.97
	CP-4	78.57	67.89	**50.49**	**57.02**	51.63	**96.37**
	CP-7	**84.16**	**72.72**	48.65	46.89	-	-
	UCW-4	66.61	63.75	40.34	55.22	**54.87**	92.26
	UCW-7	76.30	72.17	49.56	47.40	-	-

Table 3: The testing accuracy (in %) of various tokenization methods and models on 6 different downstream tasks.

Model	Token	Sequence Classification				Token Classification	
		Com	Emo	Gen	Dan	Vel	Mel
MIDI-Bert	CP-4	78.57	67.89	50.49	57.02	51.63	96.37
	CP-7	84.16	72.72	48.65	46.89	-	-
	UCW-4	66.61	63.75	40.34	55.22	54.87	96.26
	UCW-7	76.30	72.17	49.56	47.40	-	-
NG-Midiformer	CP-4	83.33	65.87	53.89	61.30	**59.82**	**96.92**
	CP-7	85.94	70.63	65.63	47.62	-	-
	UCW-4	90.63	73.44	56.67	55.05	54.13	92.31
	UCW-7	**90.63**	**81.25**	**65.87**	**63.19**	-	-

Ablation Study

In this section, we will examine the effects of unsupervised compoundation, N-Gram Transformer encoding and pre-training with N-gram respectively.

Effect of Unsupervised Compoundation Unsupervised compoundation, a tokenization method lying between REMI and CP, was evaluated for its impact on symbolic music understanding. Using the same structures as MIDI-Bert, we compared five tokenization methods on downstream tasks. As Table 2 reveals, CP-4 and CP-7 outperformed others. The diminished performance with UCW-4 or UCW-7 suggests their heavy reliance on N-gram encoding. MIDI-BERT might expect a different granularity level than what UCW provides, leading to inefficiencies in capturing the nuances of music sequences. For a fair comparison with CP, we standardized sequence lengths to 512 CPs. However, this meant only the first 512 UCW tokens were utilized, potentially affecting performance. This underscores the significance of N-Gram feature injection for model enhancement.

Table 4: The testing accuracy (in %) of on different methods for initializing downstream tasks' models.

Model	Token	Sequence Classification			
		Composer	Emotion	Genre	Dance
Midi-Bert	UCW-7	76.30	72.17	49.56	47.40
NG-Midiformer	UCW-7	**90.63**	**81.25**	**65.87**	**63.19**
w/o N-Gram Pre-training	UCW-7	58.73	68.97	31.03	52.63

Effect of N-gram Transformer Encoding We evaluated the N-gram Transformer Encoder against MIDI-Bert using four tokenization methods, as shown in Table 3. The results highlight the N-gram's advantage in music modeling, especially when combined with UCW tokenization in our NG-Midiformer. However, the two CP methods didn't benefit from N-gram in emotion classification, possibly due to lost correlations during music event merging, and N-gram encoding might not be sufficient to recover them, especially if they arise from longer-range dependencies.

Effect of pre-training with N-gram To evaluate the effectiveness of using N-gram information for pre-training, we conducted experiments to assess the model's performance on downstream tasks. We excluded the N-gram pre-training stage and present the results of these experiments in Table 4. When comparing Midi-Bert with our NG-Midiformer, we found that our NG-Midiformer results far exceeded Midi-Bert under the same input, indicating the importance of N-Gram feature injection.

However, without its pre-training, the N-Gram Transformer's performance, relying on mere random parameter

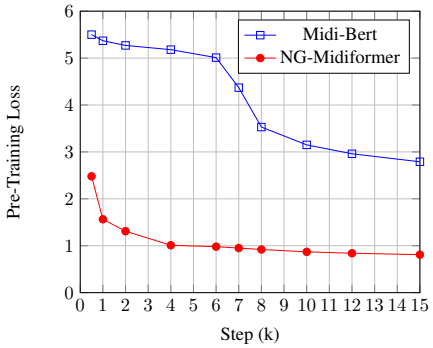

Figure 3: The convergence rate of the Bert model and our model (Taking UC-4 as an example).

updates during fine-tuning, fell below Midi-Bert, underscoring pre-training's crucial role. Additionally, with consistent hyperparameters and pre-training datasets, our N-gram Transformer encoder converged more rapidly than Midi-Bert, indicating our model's heightened efficiency in interpreting symbolic music sequences (Figure 3).

Conclusion

In this paper, we introduce the NG-Midiformer model, which proposes an unsupervised compoundation and N-gram feature injection approach. Due to the strong correlation between neighbouring elements in music, N-gram is particularly suitable for symbolic music understanding and enhances the model's ability to understand symbolic music. Our method first reconstructs the input token sequence through unsupervised methods, then pre-trains the model on a large-scale corpus using N-gram information to enhance music understanding, and finally fine-tunes on different downstream tasks to complete specific tasks. The experimental results and analysis of our method demonstrate the effectiveness of our approach. In the future, we will apply our proposed method to music generation tasks to generate various styles and genres of music.

References

Allwright, J. 2003. ABC version of the Nottingham Music Database. https://abc.sourceforge.net/NMD/index.html.

Briot, J.-P.; HADJERES, G.; and Pachet, F. 2019. *Deep Learning Techniques for Music Generation - A Survey*.

Brown, P. F.; Della Pietra, V. J.; deSouza, P. V.; Lai, J. C.; and Mercer, R. L. 1992. Class-Based *n*-gram Models of Natural Language. *Computational Linguistics*, 18(4): 467–480.

Casey, M. A.; Veltkamp, R.; Goto, M.; Leman, M.; Rhodes, C.; and Slaney, M. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4): 668–696.

Chou, Y.-H.; Chen, I.; Chang, C.-J.; Ching, J.; Yang, Y.-H.; et al. 2021. MidiBERT-piano: Large-scale pre-training for symbolic music understanding. *arXiv preprint arXiv:2107.05223*.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. 2978–2988.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Diao, S.; Bai, J.; Song, Y.; Zhang, T.; and Wang, Y. 2020. ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4729–4740. Online.

Ferreira, L.; Lelis, L.; and Whitehead, J. 2020a. Computer-generated music for tabletop role-playing games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 59–65.

Ferreira, L. N.; Lelis, L. H.; and Whitehead, J. 2020b. Computer-Generated Music for Tabletop Role-Playing Games.

Floridi, L.; and Chiriatti, M. 2020. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30: 681–694.

Hawthorne, C.; Stasyuk, A.; Roberts, A.; Simon, I.; Huang, C.-Z. A.; Dieleman, S.; Elsen, E.; Engel, J.; and Eck, D. 2019. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *International Conference on Learning Representations*.

Hirai, T.; and Sawada, S. 2019. Melody2vec: Distributed representations of melodic phrases based on melody segmentation. *Journal of Information Processing*, 27: 278–286.

Hsiao, W.-Y.; Liu, J.-Y.; Yeh, Y.-C.; and Yang, Y.-H. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 178–186.

Huang, Y.-S.; and Yang, Y.-H. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, 1180–1188.

Hung, H.-T.; Ching, J.; Doh, S.; Kim, N.; Nam, J.; and Yang, Y.-H. 2021. EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation. In *Proc. Int. Society for Music Information Retrieval Conf.*

Jackendoff, R. 2009. Parallels and nonparallels between language and music. *Music perception*, 26(3): 195–204.

joann8512. 2021. joann8512/Pianist8: First release of Pianist8.

Kong, Q.; Li, B.; Chen, J.; and Wang, Y. 2020. GiantMIDI-Piano: A large-scale MIDI dataset for classical piano music. *Trans. Int. Soc. Music. Inf. Retr.*, 5: 87–98.

Kong, Q.; Li, B.; Song, X.; Wan, Y.; and Wang, Y. 2021. High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29: 3707–3717.

Kudo, T.; and Richardson, J. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. Brussels, Belgium: Association for Computational Linguistics.

Li, T.; and Ogihara, M. 2006. Toward intelligent music information retrieval. *IEEE Transactions on Multimedia*, 8(3): 564–574.

Li, Z.; Gong, R.; Chen, Y.; and Su, K. 2023. Fine-Grained Position Helps Memorizing More, A Novel Music Compound Transformer Model with Feature Interaction Fusion.

Li, Z.; Zhang, Z.; Zhao, H.; Wang, R.; Chen, K.; Utiyama, M.; and Sumita, E. 2021. Text compression-aided transformer encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7): 3840–3857.

Liu, S.; Xu, H.; and Xu, K. 2022. An Optimized Method for Large-Scale Pre-Training in Symbolic Music. In *2022 IEEE 16th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 105–109. IEEE.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013a. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Nam, J.; Choi, K.; Lee, J.; Chou, S.-Y.; and Yang, Y.-H. 2018. Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach. *IEEE signal processing magazine*, 36(1): 41–51.

Oore, S.; Simon, I.; Dieleman, S.; Eck, D.; and Simonyan, K. 2020. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32: 955–967.

Patel, A. D. 2003. Language, music, syntax and the brain. *Nature neuroscience*, 6(7): 674–681.

Rae, J. W.; Potapenko, A.; Jayakumar, S. M.; and Lillicrap, T. P. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Sari, Y.; Vlachos, A.; and Stevenson, M. 2017. Continuous N-gram Representations for Authorship Attribution. In Lapata, M.; Blunsom, P.; and Koller, A., eds., *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 267–273. Valencia, Spain: Association for Computational Linguistics.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Shafiq, M. Z.; Khayam, S. A.; and Farooq, M. 2008. Embedded malware detection using markov n-grams. In *International conference on detection of intrusions and malware, and vulnerability assessment*, 88–107. Springer.

Sturm, B. L. 2013. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang*, Z.; Chen*, K.; Jiang, J.; Zhang, Y.; Xu, M.; Dai, S.; Bin, G.; and Xia, G. 2020. POP909: A Pop-song Dataset for Music Arrangement Generation. In *Proceedings of 21st International Conference on Music Information Retrieval, ISMIR*.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zeng, M.; Tan, X.; Wang, R.; Ju, Z.; Qin, T.; and Liu, T.-Y. 2021. MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 791–800. Online: Association for Computational Linguistics.

Zhang, Z.; Wu, Y.; Zhao, H.; Li, Z.; Zhang, S.; Zhou, X.; and Zhou, X. 2020. Semantics-aware BERT for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9628–9635.

Zhang, Z.; Zhao, H.; Ling, K.; Li, J.; Li, Z.; He, S.; and Fu, G. 2019. Effective subword segmentation for text comprehension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(11): 1664–1674.

APPENDIX

Datasets

Table 5 presents the dataset we used in this study. In the pre-training stage, we extracted and transformed the four datasets listed in the table, and ultimately obtained 24,979 UCW sequences that corresponded to the music midis. In the previous work, symbolic music datasets such as GIant-MIDI Piano (Kong et al. 2020), Maestro (Hawthorne et al. 2019), and Pop1K7 (Hsiao et al. 2021) are often utilized for pre-training. Additionally, ADL Piano (Ferreira, Lelis, and Whitehead 2020b), a widely used large-scale music dataset, is also incorporated for pre-training. By integrating these diverse symbolic music datasets, we aim to improve the model's generalization capabilities and expand its knowledge base. In the fine-tuning stage, we selected six different music classification tasks with 5 datasets to verify the performance of the model.

The GTZAN dataset (Sturm 2013) here is in wav format. We first convert it into MIDI files using the piano transcription system (Kong et al. 2021), which is a widely used neural network for transcribing music audios into MIDI format, and then convert them into the corresponding UCW sequences. We also collected the Nottingham dataset (Allwright 2003), a classic folk music collection, from an online source. To evaluate our model's performance, we utilized the MIDI version of this dataset specifically for the task of dance music classification. GTZAN dataset and Nottingham dataset are used for genre classification and dance classification, respectively. In line with our previous work, we utilized

Table 5: Summary of dataset usage.

Dataset	Usage	Pieces	Hours
Pop1K7(Hsiao et al. 2021)	Pre-training	1,747	108.8
ADL Piano(Ferreira, Lelis, and Whitehead 2020b)	Pre-training	11086	1775
GIantMIDI-Piano(Kong et al. 2020)	Pre-training	10855	1237
Maestro(Hawthorne et al. 2019)	Pre-training	529	84
POP909(Wang* et al. 2020)	Melody Extraction Velocity Prediction	865	59.7
Nottingham	Dance Classification	1034	19.42
Pianist8(joann8512 2021)	Composer Classification	411	31.9
EMOPIA(Hung et al. 2021)	Emotion Classification	1,078	12.0
GTZAN(Sturm 2013)	Genre Classification	1000	8.3

the POP909 (Wang* et al. 2020) dataset for tasks related to Melody Extraction and Velocity Prediction. Additionally, we employed the EMOPIA dataset (Hung et al. 2021) and Pianist8 dataset (joann8512 2021) for sentiment classification and composer classification tasks, respectively.

Velocity classification and melody extraction are tasks of token classification, whereas genre, emotion, composer and dance classification are tasks of sequence classification.

Constructing Unsupervised Compoundation Word

Consider a short sequence: ["Bar", "Beat_0", "Tempo_119", "G_M", "Pitch_71", "Duration_1080", "Velocity_90", "Pitch_69", "Duration_1560", "Velocity_90", "Bar", "D_7", "Pitch_71", "Duration_1080", "Velocity_88", "Pitch_73", "Duration_1560", "Velocity_90"], where G_M and D_7 are two different chords. Firstly, based on the concept of family, we group music events and obtain the following sequence: ["Bar", "Beat_0 Tempo_119 G_M", "Pitch_71 Duration_1080 Velocity_90", "Pitch_69 Duration_1560 Velocity_90", "Bar D_7", "Pitch_71 Duration_1080 Velocity_88", "Pitch_73 Duration_1560 Velocity_90"].The algorithm1 identifies the most frequent event pairs with a token, such as ("Pitch_71", "Duration_1080"), and merges them as a new token. After the first iteration, the sequence becomes ["Bar", "Beat_0 Tempo_119 G_M", "Pitch_71 Duration_1080", "Velocity_90", "Pitch_69 Duration_1560 Velocity_90", "Bar D_7", "Pitch_71 Duration_1080", "Velocity_88", "Pitch_73 Duration_1560 Velocity_90"]. Next, we identifies the next most frequent pairs ("Duration_1560", "Velocity_90"), and the sequences becomes ["Bar", "Beat_0 Tempo_119 G_M", "Pitch_71 Duration_1080", "Velocity_90", "Pitch_69", "Duration_1560 Velocity_90", "Bar D_7", "Pitch_71 Duration_1080", "Velocity_88", "Pitch_73", "Duration_1560 Velocity_90"]. From this, we can derive the UCW sequence and its associated vocabulary for this sequence.

Recall and Precision Results

In addition to achieving uniform classification accuracy, we also assessed the recall and precision metrics for our UCW-7 and UCW-4 models in the context of four downstream sequence classification tasks on our dataset, for reference purposes.

	Composer		Emotion		Genre		Dance	
	P	R	P	R	P	R	P	R
UCW-7	0.84	0.77	0.76	0.78	0.67	0.54	0.39	0.43
UCW-4	0.85	0.84	0.69	0.69	0.41	0.49	0.41	0.45

Table 6: The precision and recall values of NG-Midiformer on four sequence classification tasks using two types of tokens, where P represents precision and R represents recall.