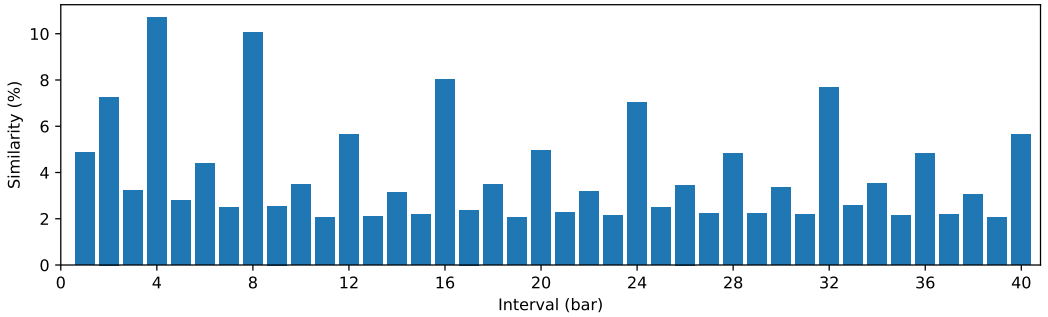
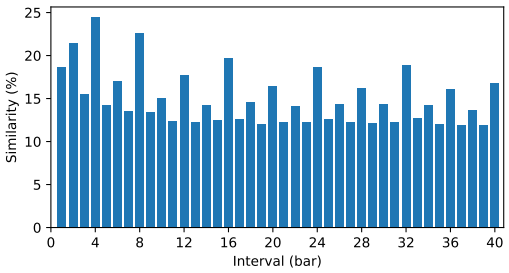
# A Similarity Statistics on Different Datasets

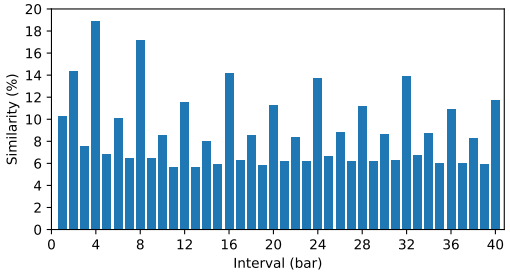
We conduct the similarity statistics introduced in §3.3 on the whole LMD dataset that we use, and exhibit the results in Figure 6. We can observe that the periodical structure pattern, that a music bar tends to be more similar to its previous 2 bars, and also to the previous $4$-th bar or its multipliers in most cases, is also satisfied on this set of music. Specifically, this rule holds on all of the instrument tracks except the drum track. This makes sense because the percussive instruments usually play the same rhythmic patterns over and over throughout a song.



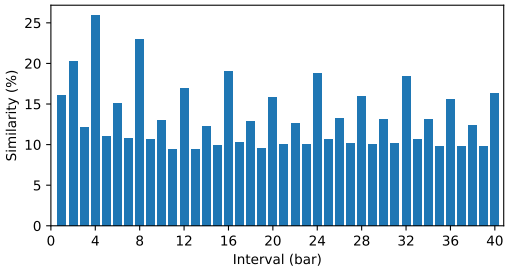

(a) Melody track.



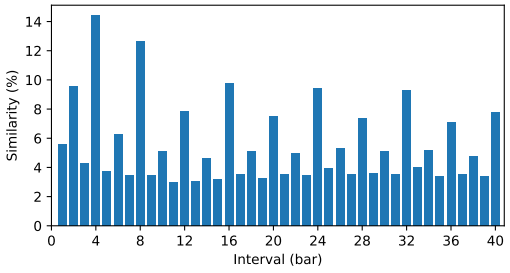

(b) All tracks.



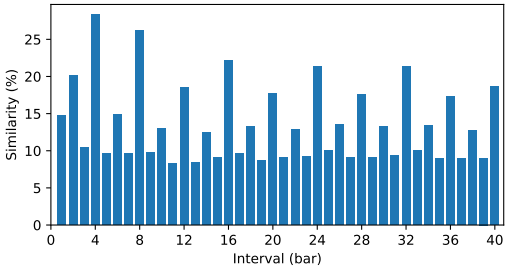

(c) Piano track.



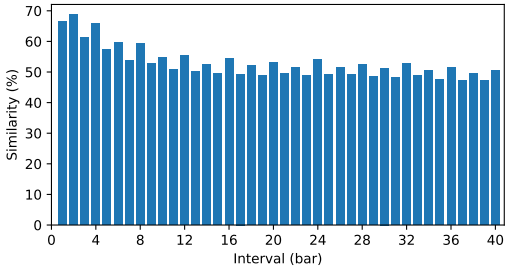

(d) Guitar track.




(e) String track.



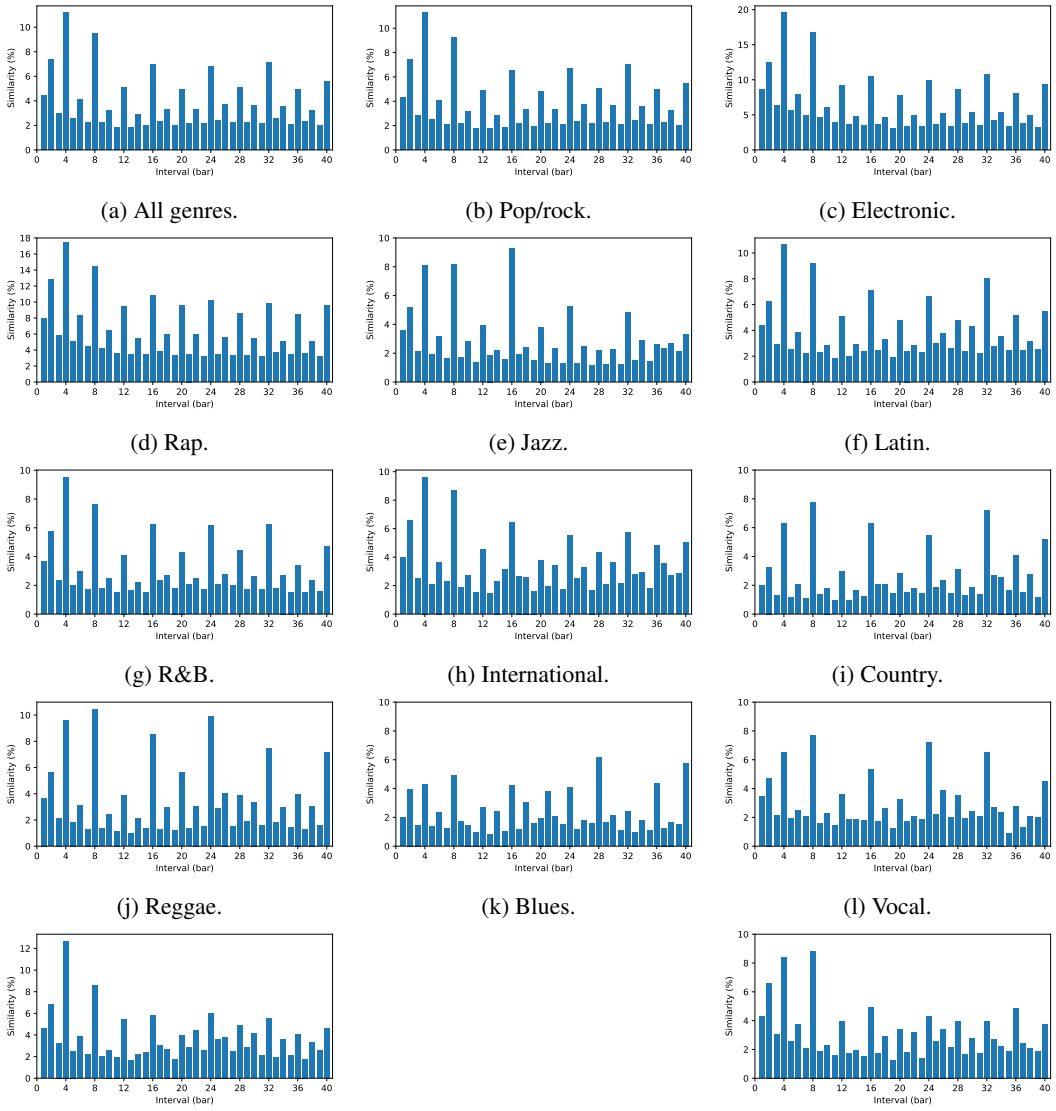

(f) Bass track.




(g) Drum track.

Figure 6: The similarity distribution of the LMD dataset we use.

To see the structure pattern of music of different genres, we conduct the same similarity statistics on the Top-MAGD dataset [6], which annotates altogether 13 music genres to the songs. Figure 7 shows that although different genres have their specific distributions, the general pattern is still applicable to all of these genres.




(a) All genres. (b) Pop/rock. (c) Electronic.

(d) Rap. (e) Jazz. (f) Latin.

(g) R&B. (h) International. (i) Country.

(j) Reggae. (k) Blues. (l) Vocal.

(m) Folk. (n) New age.

Figure 7: The similarity distribution of the melody track of the different genres in TopMAGD.

To see whether the pattern still holds on other styles of music, we conduct the statistics on the Symphony dataset [42] and exhibit the distribution in Figure 8. Since it is not easy to tell the melody tracks for the symphony music, the reported result is calculated over all the tracks. Due to the existence of some instruments that play more repetitions, e.g., drums, the differences among the similarities over the intervals are not that significant, but it still presents the same tendency.

[6] `http://www.ifs.tuwien.ac.at/mir/msd/TopMAGD.html`

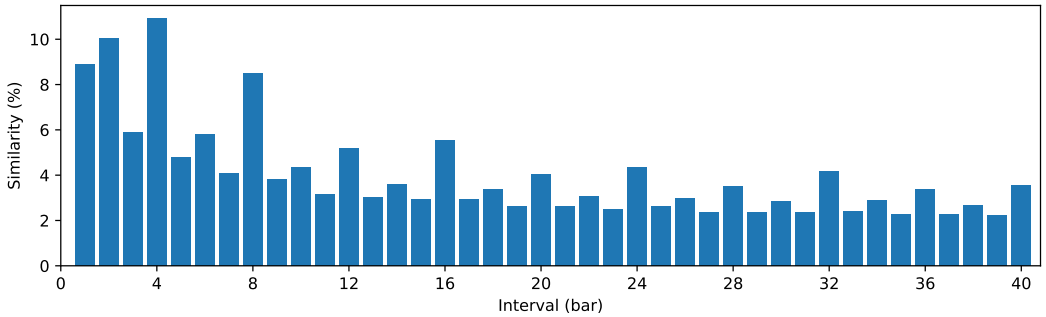



Figure 8: The similarity distribution of all tracks of the Symphony dataset.

# B Details of Experiment Settings

## B.1 Dataset Construction

We use the LMD dataset [40] in our experiments, and perform the following cleaning and processing to ensure the data quality:

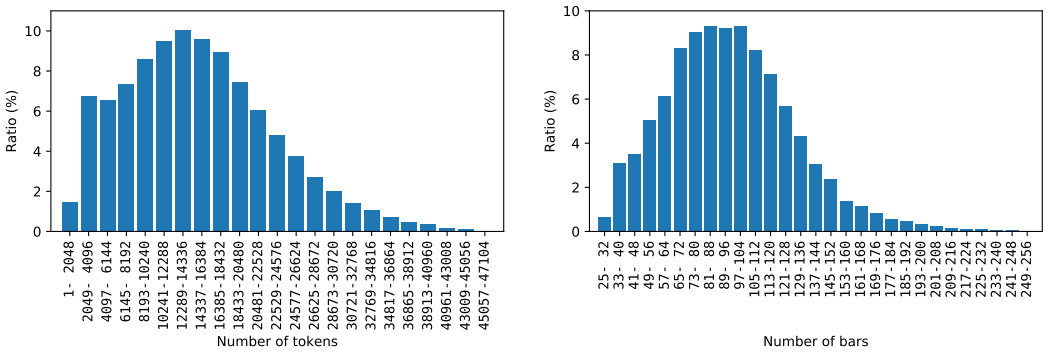
- **Track Compression**: We first compress various tracks into 6 tracks [10], namely square synthesizer, piano, guitar, string, bass, and drum, with the square synthesizer playing the melody.
- **Note Position and Duration Normalization**: To ensure that the bar splitting is correct and the duration of notes recorded in MIDI files conforms to the musically perfect duration (e.g., quarter notes held for exact 1 beat, 8-th notes held for exact 0.5 beat), we use MuseScore[7] to normalize the note position and duration.
- **Data Filtering**: Since this dataset is crawled from the Internet and contains many samples of low quality, we use a set of heuristic rules presented in Table 4 to filter them out and keep the samples of good quality and reasonable lengths.
- **Pitch Normalization**: We normalize the pitches to transfer the tonality to "C major" or "A minor".

Table 4: Our filtering rules.

| Type | Rule | Purpose |
|---|---|---|
| Duplication | Remove the duplicated samples that have the same duration and the same numbers of bars, notes, distinct note positions, and instruments. | To remove the duplicated samples. |
| Musical features | Only keep the samples of time signature 4/4.<br>Only keep the samples that have at least 2 instruments and have the square synthesizer (the melody track).<br>Only keep the samples whose tempo values (the performance speed) are not less than 24 and not larger than 200.<br>Only keep the samples whose pitch values are not less than 21 (A0) and not larger than 108 (C8).<br>Only keep the samples with maximum note duration not longer than 16 beats (4 bar).<br>Remove the samples that contain 4 or more empty bars, or the pitch/duration values are the same for all the notes. | To remove musically complicated or erroneous samples. |

We then represent each MIDI file into a sequence of tokens using a REMI-like [21] method. The bar lines are inferred automatically based on the time signature and the note onset positions. The statistics of the number of tokens and the number of bars are shown in Figure 9. The average number

[7] `https://musescore.org/`




(a) Statistics of number of tokens.

(b) Statistics of number of bars.

Figure 9: Length statistics. The vertical axes represent the ratios of those samples that are in the corresponding ranges.

of tokens is 15,042, the average number of bars is 95, and the average number of tokens per bar is 158. Most samples are longer than 10,000 tokens, making it essential to design Museformer for efficiently modeling the long music sequences.

### B.2 Implementation Details

Museformer is equipped with a dynamic sparse attention mechanism, which requires constructing distinct attention layouts for each sample according to the token ranges of bars. Since the dynamic sparse attention layout is not regular as sliding windows [14, 25] or fixed-length blocks [24], it is very challenging to leverage the GPU parallel computation techniques to speed up training and decrease the memory consumption. Note that we cannot pad each bar into a fixed-length sequence because the lengths of bars vary drastically, and it would introduce a lot of padding tokens that lead to unacceptable sequence lengths. To achieve an efficient implementation, we utilize a blocksparse method that splits the attention layouts into fixed-size square blocks, and only computes those blocks where there is at least one query-key/value pair that expects computation. In our implementation, all the summary tokens of the input bars are put before the music tokens to facilitate the computation, and thus the summarization step and the aggregation step in FC-Attention can be transferred into computing *summary-to-all* (summary tokens attending to summary tokens and music tokens) and *music-to-all* (music tokens attending to summary tokens and music tokens) attention, respectively, and their attention layouts are shown in Figure 10. We take the following 3 steps to compute each of the two attention processes: 1) *attention layout generation* to generate the full attention layout according to the bar ranges; 2) *layout block-sparsification* to transfer the full layout into a blocksprase layout; 3) *blocksparse computation* to compute the blocksparse multiplications and softmax operations.

**Attention Layout Generation** According to the bar splitting of each sample, we fill "true" on the corresponding areas of a Boolean tensor to construct the attention layout (shown on the left part of Figure 10). To speed up the construction, we collect the begin and end indices of the bars ahead of time and write a CUDA kernel to fill it for all the bars simultaneously.

**Layout Block-Sparsification** As visualized in Figure 10, the full attention layouts are block-sparsified with a fixed-size square (the block size set to 32 in our experiments), and diminished into the blocksparse layouts.

**Blocksparse Computation** We leverage SparTA [41] to compute the blocksparse multiplications and softmax operations of attention. It only computes for the shaded areas on the blocksparse layouts in Figure 10.

Since the attention layouts are the same for all the layers and heads in our setting, we only construct the attention layouts once for each sample, and cache them as well as the SparTA kernels for reuse, which saves a lot of memory and time.

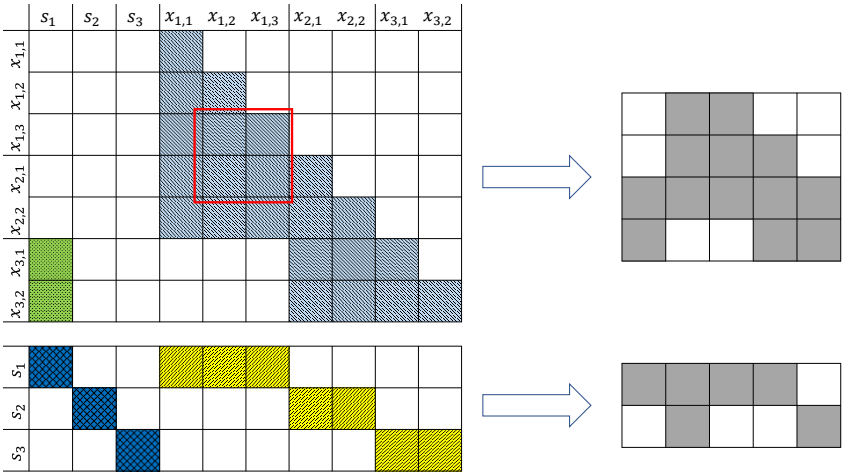



Figure 10: Visualization of the blocksparse implementation. The full attention layouts are on the left, which are exactly the same layouts shown in Figure 2b except that the summary tokens are put before the music tokens to facilitate our implementation. If the block size is 2, the corresponding blocksparse layouts are shown on the right. Note that it is only a toy example, and in real cases, the sequences are much longer, and the sparsity of the layouts is much larger than what is depicted here.

### B.3 Detailed Model and Training Configurations

For Museformer and all the compared models, the basic model and training configurations introduced in §4.1 are set the same. For Museformer involves two separate attention processes (summarization and aggregation), to make the parameter size comparable with other models, the projection matrices for projecting the targets (i.e., queries) are shared for the two attention processes, and we add different biases for summary tokens and music tokens respectively. It is the same for projecting the sources (i.e., keys and values), except that we use different projection parameters for $\tilde{\boldsymbol{S}}_{N(i)}$ in Equation (3). All the models have a comparable amount of trainable parameters: Museformer 16.1M, Music Transformer 16.6M, Transformer-XL 13.9M, Longformer 15.3M, Linear Transformer 13.2M, with acceptable differences due to different architectures and implementations.

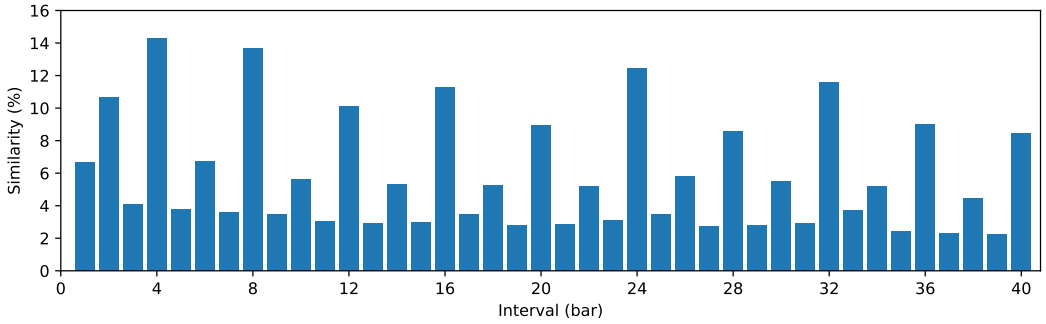
All the models are trained on 4 Nvidia V100 32GB GPUs with fp16. Since in Museformer, a token directly attends to the tokens of 8 previous bars (the selected structure-related bars) and the current bar via the fine-grained attention, for a fair comparison, we set the window sizes for Longformer to $1{,}408$, which is approximately the number of tokens that 9 bars contain. For Music Transformer that uses full attention and cannot process long sequences at once, we chunk the input sequence into blocks of fixed size $1{,}408$, and manipulate the batch size and the update frequency to ensure that it is updated the comparable number of times within each epoch as other models. For Transformer-XL, the chunk size and the memory size are also set to $1{,}408$. During inference, all the models are applied to generate sequences not shorter than $2{,}048$ and not longer than $20{,}480$, until the end-of-sentence token is generated. The top-$k$ sampling is used, and $k$ is set to 8.

## C Similarity Distributions of Generated Music

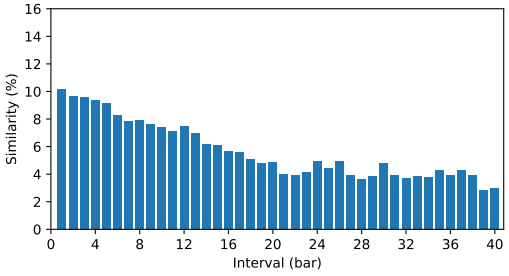
Compared to other models (the compared models and the ablation settings), the music generated by Museformer has the similarity distribution most similar to that of the training data, as its SE reported in Table 1 is the smallest. However, the value of SE may not comprehensively represent the structural characteristics, so we further display and discuss about the specific distributions in this section.

Figure 11 shows the similarity distributions. We can observe that: 1) The distribution of Museformer is very similar to that of the training data (shown in Figure 3). The quantity is close, and the contour shows the same periodical pattern, i.e., the previous 2 bars, the previous 4-th bar as well as its multipliers, have relatively large similarities. 2) The distributions of Music Transformer and Linear Transformer show no periodical pattern. It implies that the Music Transformer model trained on short sequences cannot generate well-structured music of long lengths, and Linear Transformer cannot well

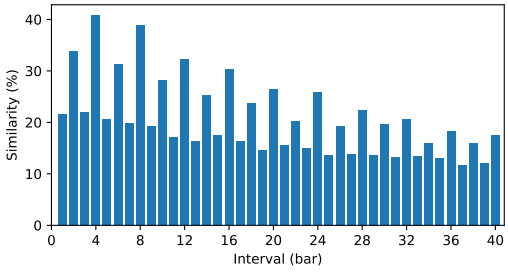
capture the structure-related correlations even though its receptive field covers the whole sequence. 3) The distributions of Transformer-XL and Longformer show the tendency of the periodical pattern, and the similarity decreases as the interval increases in general. It indicates that the two models whose receptive fields only contain the most recent content have the ability to generate periodical repetitions of short distances but fall short for long-term structures. 4) It seems that compared to the quantity of the similarity, the contour (i.e., the periodical pattern) is more relevant to the human scoring on the structure-related metrics. The distributions of Transformer-XL and Longformer show the pattern, and their subjective scores on short-term and long-term structures (shown in Table 2) are relatively high, and are higher than Music Transformer and Linear Transformer whose distributions fail to show the periodical pattern. However, the quantity of the similarity may also influence human decisions. For example, the quantities of the similarities of Transformer-XL are relatively high, which indicates that too many repetitions are generated. For these samples, human scorers sometimes think them annoying and eventually give lower scores on musicality. 5) The ablation setting, Museformer w/o coarse-grained, has slightly larger SE compared to Museformer, and its distribution clearly shows the periodical pattern. Thus, the coarse-grained attention does not contribute a lot to the music structures. The distribution of Museformer w/o bar selection shows the tendency of the periodical pattern, and the similarity decreases in general as the interval increases, similar to those of Transformer-XL and Longformer. It indicates that the structure-related bars are contributory to generating music with both short-term and long-term structures.



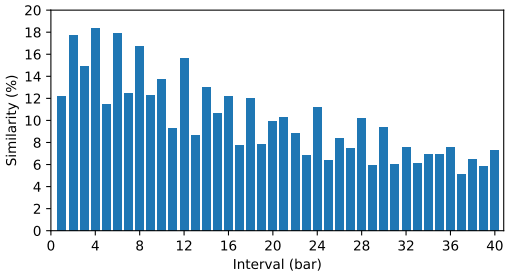

(a) Museformer.



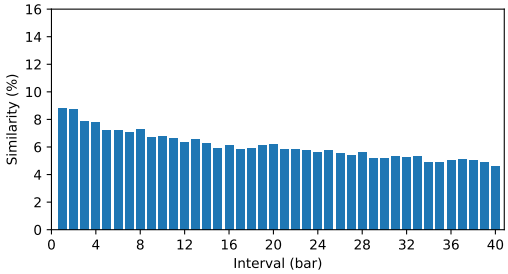

(b) Music Transformer.



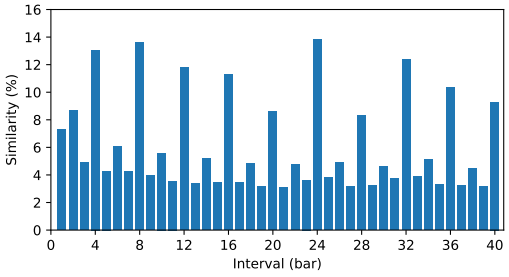

(c) Transformer-XL.



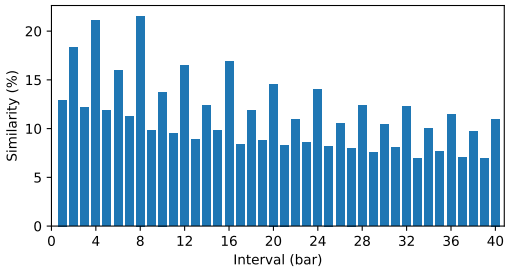

(d) Longformer.




(e) Linear Transformer.




(f) Museformer w/o coarse-grained.




(g) Museformer w/o bar selection.

Figure 11: The similarity distribution of the melody track of the music generated by different models.