BeMuse: The Major Key to Symbolic Retrieval

Elliott Zackrone
University of Washington
ezackr@cs.washington.edu

Aruna Srivastava
University of Washington
arunasri@cs.washington.edu

Rei Tao
University of Washington
taocla17@cs.washington.edu

GitHub: https://github.com/ezackr/BeMuse/tree/main

1 Introduction

Musical scores explicitly encode the fundamental melodic, harmonic, and rhythmic elements that define a composition, in contrast to audio representations that capture surface-level acoustical qualities. For musicians seeking to identify the origin of a musical excerpt or melody, symbolic score-based retrieval may offer more accurate results compared to audio-based approaches. Furthermore, existing tools in audio-based retrieval demonstrate limitations with noisy audio input. Motivated by the constraints of audio-based search engines like Shazam, we introduce BeMuse: a novel score-based melody search engine to allow musicians to provide symbolic music queries.

2 Dataset

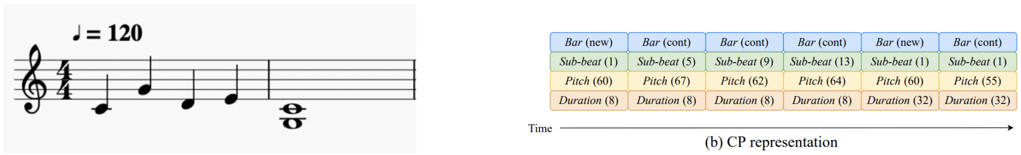
For this paper, we utilize the mono-midi-transposition dataset, which is derived from the mono-MusicXML dataset (1). The mono-MusicXML dataset consists of monophonic musical pieces uploaded to MuseScore, a popular online platform for creating and sharing sheet music. The mono-midi-transposition dataset will provide MIDI files of monophonic pieces and introduces variation by randomly transposing each piece into a new key signature. This augmentation step helps to increase the diversity of the dataset and mitigates potential biases that may arise from the original key signatures.

It is worth noting that this dataset is limited to monophonic scores; While this helps us focus on the performance of our model, our findings may not be generalizabe to polyphonic scores.

3 Methods

3.1 Pre-Processing

To create compound words of a given Midi file, we pre-process MIDI sequences into 4-tuples such that a note can be defined by its given bar, position, pitch, and duration. Duration and position of a note is quantized into 16 sub-beat divisions, in terms of 64th notes, where each sub-bar division represents a fraction of a beat. The maximum duration allowed by MidiBERT is roughly 4 beats and pitch range is between 27-112, or 0-85 which spans approximately 6 octaves from low A and extending to high C (2). This pre-processing handles various aspects of MIDI file parsing, including time signature changes, bar and position computations, and pitch and duration normalization to comply with MidiBERT's input format constraints. We use a padding function to ensure consistent sequence lengths of 512 tuples, facilitating batch processing and efficient training with MidiBERT.


♩ = 120
Bar (new) | Bar (cont) | Bar (cont) | Bar (cont) | Bar (new) | Bar (cont)
Sub-beat (1) | Sub-beat (5) | Sub-beat (9) | Sub-beat (13) | Sub-beat (1) | Sub-beat (1)
Pitch (60) | Pitch (67) | Pitch (62) | Pitch (64) | Pitch (60) | Pitch (55)
Duration (8) | Duration (8) | Duration (8) | Duration (8) | Duration (32) | Duration (32)
Time

(b) CP representation

Figure 1: sequence of notes and the respective compound word representation for MidiBERT input

3.2 MidiBERT

To produce embeddings, we use MidiBERT which uses symbolic domain melody extraction for note-level classification. This allows for suitable representation that can be compared with the encoded representations of the musical scores in the database. MidiBERT will serve as an encoder for the 4-tuples extracted during preprocessing, transforming the MIDI data into high-dimensional vector representations which are then flattened through a linear transformation. These scalars now capture musical context and structure which are suitable for training.

3.3 Evaluation

3.3.1 DAMP Dataset

For the evaluation of our research, we employ the Stanford Densely Annotated Multitrack Precision (DAMP) dataset. The DAMP dataset is a collection of high-quality song recordings, along with their corresponding karaoke versions, where the lead vocals have been removed and slight variations in pitch, duration, and rhythm from the original pieces can be heard. Though a formal analysis was not performed, we queried some tracks from the DAMP dataset to understand where the model performance degradation originates from.

The unique aspect of the DAMP dataset is that both the original song audio and the karaoke audio are preprocessed and converted into MIDI format, allowing for direct comparison and evaluation of symbolic music representations.

3.3.2 Assesment

During evaluation we encode both the actual song MIDI and the karaoke MIDI into high-dimensional vector representations using our trained model. These vector representations capture the intrinsic musical features and structures present in the respective MIDI files. For each karaoke MIDI vector, we computed its similarity scores against all the actual song MIDI vectors in the dataset. This was achieved through contrastive loss. We then checked if the actual song vector corresponding to the current karaoke MIDI vector appeared within the top-5 most similar vectors. If the actual song vector was among the top-5 retrieved results, it was considered a successful retrieval.

4 Experiments

4.1 Augmentation

Transposition augmentation randomly shifts all notes up or down by a constant value. Rhythm augmentation doubles or halves the length of a given MIDI such that a piece that may be 4 bars long can be halved to be 2 bars long where eighth notes replace instances of quarter notes. Accidental augmentation is applied to each individual note in a track where a note's pitch is shifted slightly up or down by some probability p. We will experiment with combinations of different types of augmentations to determine which method creates more robust data that improves the models performance.

4.2 Training

During the training process, we employ the contrastive loss function. Contrastive loss is defined pair-wise for a dataset $\{(x_i, y_i)\}_{i=1}^{N}$ where, for our use case, x_i and y_i are derived from the same original song (e.g. karaoke and original midi). Each pair (x_i, y_i) are encoded as high dimensional vectors $(\hat{x}_i, \hat{y}_i)$. Contrastive loss attempts to maximize the vector similarity for pairs $\hat{x}_i, \hat{y}_i$ for the same i, and minimize vector similarity for pairs $\hat{x}_i, \hat{y}_j$ where $i \neq j$. This formulation pushes the model to encode vectors from the same original midi into similar position in latent space. Otherwise, the model is encouraged to map inputs from different midis into varying position in latent space, by pushing the dot product towards 0.

By minimizing this contrastive loss, the model learns to map positive pairs (actual song and karaoke MIDI) to similar representations in the high-dimensional vector space, while pushing negative pairs (unrelated MIDI files) apart. Given two songs of the same base song, this process aims to produce vectors with a high similarity score. This training objective enables the model to effectively capture and encode the intrinsic musical similarities and differences between symbolic representations, facilitating accurate retrieval during the evaluation phase.

5 Results

1) Augmentation

Transposition	Rhythm	Accidentals	Dataset Size	Performance
1	-	-	11688	58%
2	-	-	17532	65%
2	1	-	35064	64%
2	-	1	35064	69%
3	-	2	70128	74%

Table 1: Performance with Different Augmentations

Across various augmentation methods, we find that a single transposition results in a relatively poor performance at 58%. To improve this initial performance and robustness, we test various methods of augmentation in addition to transpositions, including rhythm-based augmentation and additional accidentals.
We calculate the total number of songs in the training data as

$$\text{songs} = (\text{base}) \cdot ((\text{transpositions} + 1) \cdot (\text{accidentals} + 1))$$

Adding an additional transposition to our initial transposition increases the dataset size by 5844 MIDI songs and improves top-5 performance by 7%, suggesting that transpositions, while expensive, is an effective augmentation method.

Other augmentation techniques prove less effective; Initially, we believed rhythm based augmentation to prove effective as we expected query and song pairs to differentiate in duration. Yet, adding an additional rhythm augmentation with two transpositions increased our training size from 17532 to 35064 but decreased performance by 1%. Due to the inherit nature of Karaoke, singers are guided in duration and speed when singing, such that rhythm is not the field that causes singing audio to deviate from the actual song. Rather, it is pitch that is most prone to variation in between query and song pairs.

We begin using accidentals to increase training data diversity and find that a single accidental with two transpositions at the same size of 35,064 files increases performance by 4%

With 3 transpositions and 2 accidentals, we obtain a 74 % accuracy for top-5 accuracy, just short of the 80% baseline achieved for top-1 accuracy.

6 Discussion

The results of our experiments highlight the effectiveness of BeMuse, our proposed symbolic music search engine, in retrieving relevant musical scores based on melody queries. By leveraging the transformer-based MidiBERT model and carefully crafted data augmentation techniques, we were able to achieve promising retrieval accuracy on the Stanford DAMP dataset.

One of the key findings from our study is the importance of data augmentation in enhancing the model's robustness and generalization capabilities. The transposition and accidental modifications proved to be particularly effective, as they introduced variations in pitch. These augmentations helped the model better handle the deviations between singing performances and original recordings, leading to improved retrieval accuracy.

Furthermore, our evaluation performed with the DAMP dataset, showcased the practical applicability of BeMuse in a realistic scenario. The achieved top-5 retrieval accuracy of 74% demonstrates the model's ability to capture and retrieve semantically similar music representations, even in the presence of variations introduced by vocal performances.

With additional resources and time, we believe further augmentation, specifically with transpositions and accidentals, will allow for improved performance and top-1 performance. Current compute limitations prevent large amounts of augmentation from being integrated but could be a point of exploration for the future. Other future work may include taking audio input or polyphonic symbolic input.

7 Conclusion

This research presented BeMuse, a symbolic music search engine that enables retrieving original musical scores from melody queries. By leveraging MidiBERT, a transformer model for music data, we encoded MIDI files into high-dimensional vectors capturing musical context and structure. Extensive data augmentation techniques, including transposition and accidental modifications, were employed to enhance model robustness. Evaluated on the DAMP dataset of real song recordings and corresponding karaoke versions, BeMuse achieved a promising 74% top-5 retrieval accuracy. Our approach demonstrated effective utilization of rich symbolic music representations for accurate retrieval, even with variations introduced by vocal performances. BeMuse represents a significant step toward bridging the gap between symbolic formats and music information retrieval systems, enabling more efficient discovery and exploration of musical works.

References

[1] S. Garcia-Valencia, A. Betancourt, and J. G. Lalinde-Pulido, "Sequence generation using deep recurrent networks and embeddings: A study case in music," *ArXiv e-prints*, vol. abs/2012.0, 2020. [Online]. Available: https://arxiv.org/abs/2012.01231

[2] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "Midibert-piano: Large-scale pre-training for symbolic music understanding," 2021.