# 修 士 論 文 の 和 文 要 旨

| | | | |
|---|---|---|---|
| 研究科・専攻 | 大学院　情報理工学研究科　　情報・ネットワーク工学専攻　博士前期課程 | | |
| 氏　　名 | Mastuti Puspitasari | 学籍番号 | 2131179 |
| 論 文 題 目 | SBERT-based Musical Components Estimation from Lyrics Trained with Imbalanced Data<br>(不均衡データで学習するSentence-BERTに基づく歌詞からの音楽部品推定の研究) | | |


要　　旨

本研究では、自動作曲において歌詞入力から音楽要素設定が自動化されるモデルを検討し提案する。このモデルの出力には、和声進行・リズムパターン・楽器・テンポ・ドラムパターンの歌詞と相関性の高い5つの音楽特徴が含まれる。自動作曲システムを使用する場合には、このような音楽特徴セットを使用して、自分の音楽を作曲したい場合の推奨値を歌詞から自動で得ることができ、その推奨値により自動作曲設定ができる。

本モデルでは、「Orpheus」という自動作曲システムの過去のユーザーデータから抽出した歌詞と作曲条件の対を学習させる。ユーザーが書いた歌詞とその歌詞に対応するユーザーが選択した5種類の音楽特徴のサンプル数は24,000件を超える。

この歌詞データは、BERT（Bidirectional Encoder Representations from Transformers）の一種、「Sentence-BERT」という自然言語処理モデルで歌詞埋めベクトルに変換され、楽曲データとともにモデル学習に利用される。過学習を減らすために、学習データは音楽特徴ごとにサンプル数が最も多い種類の上位10種のサンプルを含むように制限される。それらのサンプルはフォーカル損失関数で学習された多層パーセプトロンモデルに入力される。

提案モデルは、従来モデルと比較し、それよりスコアが高いかどうかを確認するために、ROC(Receiver Operating Characteristic)とF1スコアで客観的に評価した。また、提案モデルで作曲した音楽サンプルが、元の「Orpheus」ユーザーが作曲した音楽と前モデルで作曲したサンプルと比較し、歌詞合ってる曲を作曲できるかどうかを確認するために、アンケート調査を実装した。上記の2つの評価方法で、提案モデルが従来モデルよりも、いくつかのケースで元の作曲者、「Orpheus」ユーザーよりも高いスコアを得られるということがわかった。

令和５年度　修士論文

# SBERT-based Musical Components Estimation from Lyrics Trained with Imbalanced Data

不均衡データで学習する Sentence-BERT に基づく

歌詞からの音楽部品推定の研究

電気通信大学大学院 情報理工学研究科

情報・ネットワーク工学専攻

2131179　Mastuti Puspitasari

指導教員

中鹿 亘 准教授

南 泰浩 教授

令和5年9月4日

# Abstract

In this research, we propose a model that can derive musical components from lyrics as input. The output of this model includes 5 musical components that are highly correlated with lyrics, which include: chord progressions, rhythm patterns, instruments, tempo, and drum patterns. Using such a set of musical components, one can get recommendations in case they want to compose their own music, or in the case of the use of automatic composition system, the recommendation can be used to automate the setup.

This model is trained with lyrics and composition conditions extracted from past user data of an automatic composition system called Orpheus. More than 24,000 samples are extracted, consisting of lyrics and the corresponding 5 musical components selected by the Orpheus users.

These lyrics are converted into lyrics embedding with Sentence-BERT, a language model variant of BERT (Bidirectional Encoder Representations from Transformers) to be used in model training together with the musical component data. To reduce overfitting, training data is limited to include the top 10 options of each musical components with the highest number of samples and these samples are fed into a Multi Layer Perceptron model trained with focal loss function.

The proposed model is evaluated objectively by comparing it with the previous model using ROC and F1 scores to see if the former can outperform the latter. A set of surveys was also done to see how well the resulting samples of the proposed model in comparison to the original composition by an Orpheus user and the samples created with the previous model. The 2 evaluation methods show that the proposed model scores higher than the previous model and sometimes the original composers, the Orpheus users themselves.

本研究では、自動作曲において歌詞入力から音楽要素設定が自動化されるモデルを検討し提案する。このモデルの出力には、和声進行・リズムパターン・楽器・テンポ・ドラムパターンの歌詞と相関性の高い 5 つの音楽特徴が含まれる。自動作曲システムを使用する場合には、このような音楽特徴セットを使用して、自分の音楽を作曲したい場合の推奨値を歌詞から自動で得ることができ、その推奨値により自動作曲設定ができる。

本モデルでは、「Orpheus」という自動作曲システムの過去のユーザーデータから抽出した歌詞と作曲条件の対を学習させる。ユーザーが書いた歌詞とその歌詞に対応するユーザーが選択した 5 種類の音楽特徴のサンプル数は 24,000 件を超える。

この歌詞データは、BERT（Bidirectional Encoder Representations from Transformers）の一種、「Sentence-BERT」という自然言語処理モデルで歌詞埋めベクトルに変換され、楽曲データとともにモデル学習に利用される。過学習を減らすために、学習データは音楽特徴ごとにサンプル数が最も多い種類の上位 10 種のサンプルを含むように制限される。それらのサンプルはフォーカル損失関数で学習された多層パーセプトロンモデルに入力される。

提案モデルは、従来モデルと比較し、それよりスコアが高いかどうかを確認するために、ROC(Receiver Operating Characteristic) と F1 スコアで客観的に評価した。また、提案モデルで作曲した音楽サンプルが、元の「Orpheus」ユーザーが作曲した音楽と前モデルで作曲したサンプルと比較し、歌詞合ってる曲を作曲できるかどうかを確認するために、アンケート調査を実装した。上記の 2 つの評価方法で、提案モデルが従来モデルよりも、いくつかのケースで元の作曲者、「Orpheus」ユーザーよりも高いスコアを得られるということがわかった。

# Table of Contents

# Chapter 1

# Introduction

This research aims to find appropriate approach that can be used to develop a system that is capable of recommending specific musical components based on lyrics input that is optimized on imbalanced training data. To explain why such research needs to be done, let us first cover the more general field of automatic music composition, in which our scope of research belongs.

## 1.1 Automatic Music Composition

Musical composition, according to [5], is the act of conceiving a piece of music, the art of creating music, or the finished product. The first definition, which is the focus of our research, is conventionally done with physical instruments. Following the development in computing technology, this process has been made possible to be done digitally. With a digital audio workstation (DAW), an application software used for recording, editing ,and producing audio files, it is now possible to compose music with no instruments. Some well-known examples of DAW include: Ableton Live [1], Image-Line FL Studio [2], and Apple Logic Pro. [3]

However, even with DAW, it is still not easy to compose music, especially for those with minimum musical knowledge. Fortunately, there are automatic composition tools that utilizes artificial intelligence (AI) as an alternative for such people. In a web-based music generation tool SOUNDRAW [4], for example, users can generate music by choosing genre, mood, and theme. They can also specify the length, tempo, and the instruments of the music they want to generate.

One downside to such tool is that the generated music is instrumental. The absence of vocal components in composition generated with AI is apparently a common occurrence, even in AI model which takes in textual prompt to generate music such as MuseNet [6]. While there is nothing wrong with not having vocal components in a composition, such output is not suitable for users who want to explicitly convey a story through lyrics.

This issue is fortunately resolved in Orpheus, a Japanese web-based automatic composition system introduced in [7] that takes in lyrics and various musical components as input and generates music based on a detailed setup as shown in Fig. 1–1. Each setup in this configuration consists of numerous options. Chord progression setup, for example, has over 1,500 unique options for users to choose from. While variety is important, detailed settings with too many options can unfortunately be overwhelming for beginners, consequently risking them giving up. The lack of feature that complements this complexity by helping users to easily grasp and quickly use the system becomes a big issue that needs to be solved.

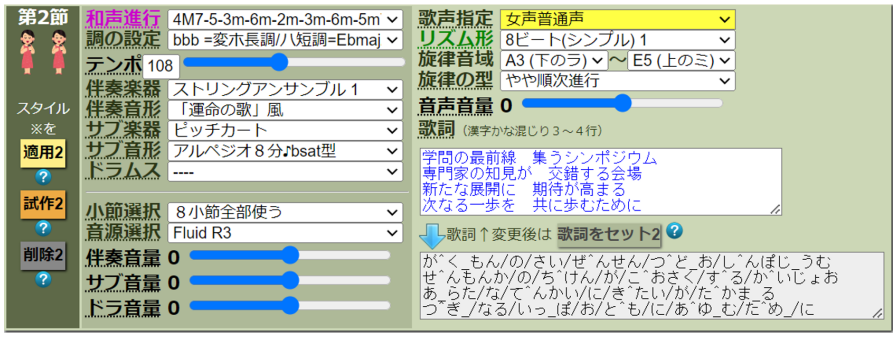


Fig. 1–1: Complex musical components and lyrics configuration in Orpheus

Just like how lyrics can be generated with textual AI such as ChatGPT [8], with a proper approach, we argue that this selection process can be automated, by providing setting recommendations to the users. While automation of this process is certainly possible with random choice, a baseless automation will likely result in a problematic composition and user dissatisfaction. As the lyrics are submitted into the system, the users should also want the generated composition to match the lyrics they have provided in terms of, for example, emotion, context, and complexity. By creating a system that can recommend several musical components based on lyrics input, we hope to simplify this selection process.

## 1.2 Automating the Selection Process

To automate the selection process in an automatic composition system such as Orpheus, there are 2 common approaches: rule-based and machine learning. A rule-based system applies human-made rules to store, sort and manipulate data, mimicking human intelligence as a result. It requires a data source and set of rules for data manipulation.

With more data added to the system, the system will become more complicated exponentially, as the rules need to cover all combinations possible with the available data. In addition to that, in-depth knowledge is necessary to specify the rules. Considering the number of data and complexity in music composing, such system may not be the best approach to solve the selection process issue, leaving us with the second approach: machine learning.

### 1.2.1 Machine Learning as a Solution

According to [9], machine learning, in artificial intelligence which is a subject within computer science, is a discipline concerned with the implementation of computer software that can learn autonomously. It uses two types of techniques: supervised learning, which trains a model on known input and output data, and unsupervised learning, which finds hidden patterns in input data.

While both are viable, AI models trained with supervised learning tend to perform better than those trained with unsupervised learning. This is reasonable as expected outputs are provided in addition to input in the training data, making it easier for the models to learn. However, this also means that each data needs to be labeled, consequently adding the time necessary to prepare the dataset. To achieve our goal with AI models trained with supervised learning, we need a dataset that consists of lyrics and the corresponding musical components, and fortunately, such data can be extracted from Orpheus itself.

### 1.2.2 Orpheus as a Source of Training Data

Since 2006, the year of its release, Orpheus has generated over 700,000 pieces of composition that were configured by more than 15,000 registered users. Around 6,000 of those pieces are published and open to view for public. It is stated on the website that users need to agree to share their composition with researchers to use, which makes it lawful to extract these data for research purpose.

A small downside to using such data for research is the fact that we have no control over user input. The extracted data is prone to be imbalanced in terms of number of samples, as there is no way to ensure that each options in the setup is equally used by the users. This results in some options having low number of composition samples. To avoid overfitting caused by training options with insufficient data, we decided to exclude those with low numbers of samples, which will be explained further in 3.2.

## 1.3 The Correlation between Music and Its Components

With a dataset available to use and a core idea on how to progress, one may question the base of this approach. After all, it is not reasonable to infer musical components from lyrics if they are not correlated to begin with. To understand the correlation between lyrics and musical components, let us first look at those that are available on Orpheus and how they correlate with the corresponding music, as it is the bigger scope in which they are a part of.

In Orpheus, there are 14 musical components available to setup. To limit the scope of this research, we picked 5 that have been proven derivable from either genre and mood, the two general characteristics of music that have been explored extensively by many researchers. The 5 musical components include: chord progression, rhythm pattern, instrument, tempo, and drum pattern.

### Chord Progression

Chord in music, as described in [10], consists of three or more single pitches heard simultaneously It may be consonant, implies repose, or dissonant, implies subsequent resolution to and by another chord. A chord progression is a sequence of chords that shows the chords in order of when they are being played in a song. According to [11], it is a fundamental building block in music which decides the overall mood of a song, as many composers would compose music starting with chord progressions followed by adding melody and details. Their work explores the effect of chord progressions in music emotion recognition and shows that chord progressions have influence in music emotion.

**Rhythm Pattern**

According to [12], rhythm is the pattern of music in time. It is the one indispensable element of all music. An experiment in [13] has shown that, while it is error-prone and not always unique to a genre, music can be classified by rhythmic pattern alone with a 50% correctness, showing its correlation with genre.

**Instrument**

As described in [14], a musical instrument is a device used to produce a musical sound. They are often classified by the method of producing sound, which includes percussion, stringed, keyboard, wind, and electronic. With each genre of music, there is a tendency to use certain musical instruments in a composition. This is proven by [15], which investigated the effect of separating music tracks and extending feature vector by parameters related to specific musical instruments in automatic musical genre classification and achieved promising results that shows the correlation between musical instruments and genre.

**Tempo**

Tempo, according to [16] is the speed or pacing of a piece of music. It plays an essential role in performance and acts as the heartbeat of expression. In a series of experiments done in the research [17], the results show that depending on the genre of the musical excerpts, there is a difference in what tempi were perceived as most salient. The result of their research shows that there is a noticeable tendency in preference of tempo based on genre.

**Drum Pattern**

In [18], drum is described as a musical instrument, which sound is produced by the vibration of a stretched membrane and a part of the larger category of percussion instruments. Drum pattern itself is the pattern of how a set of drums is played in a composition. In an attempt to develop faster and more efficient tools for music content analysis, the rhythm of drums, which is correlated with drum pattern, was used in [19] to classify music genre automatically, resulting in an effective approach as shown in their qualitative evaluation, proving the tendencies of music with the same genre to share similar drum pattern.

As it turns out, mood and genre seem to have high correlation with the 5 musical components that are all available in Orpheus. This correlation is vital in this research, and will be further discussed in 2.2.

## 1.4 Chapter Summary

The contents of each chapter are summarized as follows:

Chapter 2 discusses the related works that have provided us with a base to work on and ideas on how to achieve the goal of this research.

Chapter 3 introduces the proposed model that implements Sentence-BERT pre-trained proper corpus and focal loss function in an attempt to solve the issue taken as a research subject and better the base approach proposed in [25].

Chapter 4 shows the experiments that we have done by combining different pre-processing methods with different corpus and loss function during training and other attempts in order to find the best setup for the proposed model.

Chapter 5 provides several evaluation results which proves the better performance achieved by the proposed model in comparison to the previous model and discusses potential future research topics.

Chapter 6 concludes and summarizes the research and its findings.

# Chapter 2

# Background and Related Works

Previously, we discussed automatic music composition, the necessity of automating option settings from lyrics, the potential of Orpheus [7] as a source of training data, and the correlation between music and its components. In this chapter, we will introduce related works that were done prior to this research which further motivated us to conduct this research.

## 2.1 Studies in Automatic Composition

Prior to this research, there have been several other attempts that explore how to automate music composition with different approach which does not involve the use of lyrics. We will shortly explain how each works and discuss both the advantages and disadvantages of these approaches.

### 2.1.1 Generating Music with Similar Style

The model introduced in [20], DeepBach, was trained with the chorale harmonizations by Johann Sebastian Bach, and as a result, is capable of generating chorales in a style that is similar to that of Bach. Audio samples generated with this model is also available in [20]. It is theoretically applicable to other styles of music, as the output depends on the training data and is steerable in terms of notes, rhythms, and cadences, which provides control over the generated score.

However, this becomes their limitation for the same reasons. Separate models need to be trained for different styles with different sets of data. In addition to that, considering the constraints, musical knowledge is still necessary to some degree for users to be able to properly navigate such models.

### 2.1.2 Score-based Music Synthesis

In another research, DeepPerformer [21], a model that is capable of synthesizing audio samples based on musical score input was introduced. Their model was trained with violin and piano recording data that are paired with their corresponding musical score. Refer to [21] for audio samples generated by this model.

This approach allows users to generate music with musical scores that they have without the need of manually configure the composition. With some modifications, this model can also potentially vocalize lyrics in the musical scores. The downside of this approach is that by using score as input, usage of such models is limited to those who are music literate. The aspect of music composing itself is not covered in the automation, as it is done through making the scores, and not in the music synthesizing process. It is not going to be useful for people without the knowledge as they probably are unable to compose to begin with.

### 2.1.3 Audio Generation from Rich Captions

Lastly, in a very recent research [22] as of 2023, MusicLM was introduced. It is a model that is capable of generating complex music composition based on text input. This textual input is used as a description, which becomes the base of the generated music. Various use cases and audio samples generated by MusicLM with the corresponding prompts are available in [22].

MusicLM is also able to include vocals in their generated music, making it the closest to what we are trying to achieve in this research. Unfortunately, these vocals are of meaningless words and there is no mean of adding lyrics in the generation, which led us to search for other works that involve lyrics.

## 2.2 Lyrics and Musical Components

With some of the attempts in automating musical composition covered, let us now consider the possibility of using lyrics as input to automate music composition by looking at some studies that involves lyrics and explore the relationship between lyrics, genre, and mood. This indirect relationship between lyrics with the 5 musical components, shows the potential of estimating musical components with lyrics input. To support this claim, let us introduce a number of these works and why they are relevant in our research.

### 2.2.1 Genre Estimation with Lyrics

In [23], they proposed a model that is capable of classifying Turkish lyrics into three meta-data, including author, year of release, and genre. Their proposed model managed to achieve relatively high Receiver Operating Characteristic (ROC) score of 92.5%, proving that it is possible to infer genre from lyrics. This is supported by another work [24], which classifies Nordic lyrics into genre.

### 2.2.2 Chord Progression Estimation with Lyrics

While genre estimation is enough for simple AI tools that take in genre input to generate music, it is not enough for a more complex system with various musical components. The research [25] takes in one of those musical components, chord progressions, as a research subject to see whether it is possible to infer those from Japanese lyrics. This research also used Orpheus as a source of training data, and although their proposed model was not able to achieve decent accuracy, their approach can still be used as a base to work on in this research.

### 2.2.3 Melody-based Lyrics Generation

The research [26] proposes an end-to-end melody-conditioned lyrics generation system based on Sequence Generative Adversarial Networks (SeqGAN) as proposed in [27]. With melody input, their system is capable of generating a line of lyrics. Considering what they have achieved, it is only reasonable to assume that the opposite, the goal of our research, is also possible to achieve.

## 2.3 Lyrics Pre-processing for Machine Learning

As computers do not understand the meaning of words, it is impossible for them to process textual data such as lyrics directly. A representation method is necessary to allow computers to differentiate textual data. In [25], for example, word2vec as proposed in [28] is used to create vector representation of lyrics.

### 2.3.1 Utilizing Word2Vec to Pre-process Lyrics

Word2vec was proposed to be a model that can create vector representations of single words. The vector representation $\boldsymbol{v}_i$ for each $w_i$ as a center word in their vocabulary is calculated by processing the one hot representations of the surrounding words with the weight as shown in Fig. 2–1, which results in a vector representation with the predetermined size of $d$.

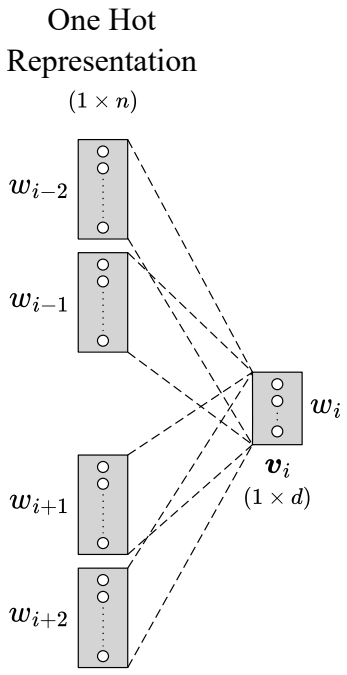


Fig. 2–1: Summation of neighboring word vectors

This model is trained by estimating one hot representation of $w_i$ from $\boldsymbol{v}_i$ and comparing it to the actual one hot representation in the vocabulary as shown in Fig. 2–2. The difference between the 2 is then used to update the weight. This strategy is known as continuous bag-of-words (CBOW).

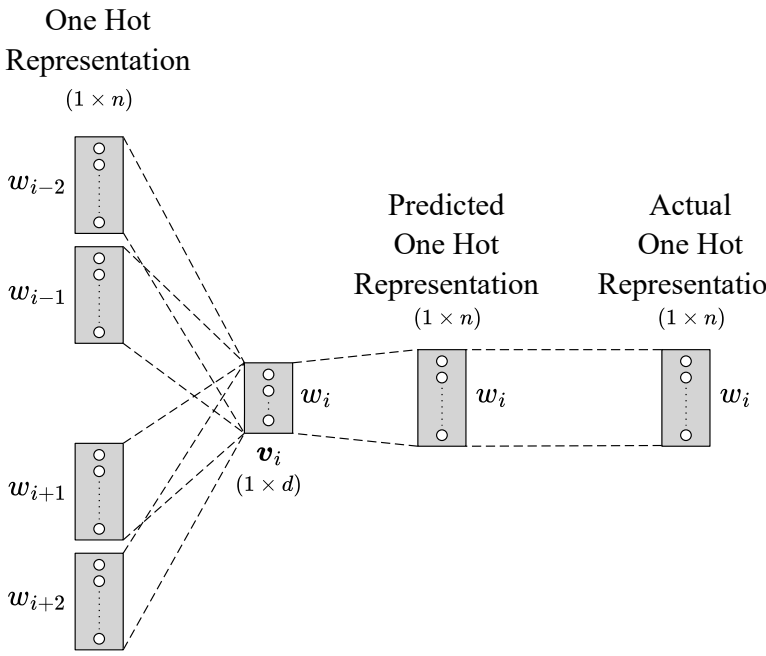


Fig. 2–2: Predicting and comparing one hot representation of $w_i$

Besides CBOW, word2vec can also be trained with another strategy known as skip-gram. While CBOW predicts a word based on its surrounding words, skip-gram does the opposite of CBOW by predicting the surrounding words given a specific word as shown in Fig. 2–3.

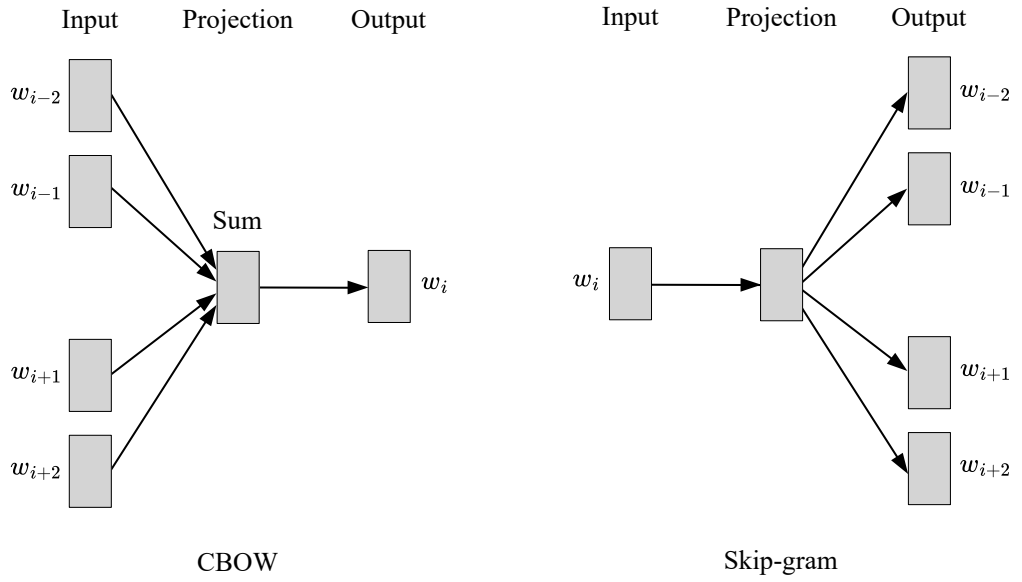


Fig. 2–3: The architecture of CBOW and skip-gram models

Word2vec assigns 2 random $d$-dimensional vectors to any word $w_i$ which represent it as a *center* and *context* word, denote by $\boldsymbol{v}_i \in \mathbb{R}^d$ and $\boldsymbol{u}_i \in \mathbb{R}^d$ respectively. For CBOW, the conditional probability of generating any center word $w_c$ with the context vector $\boldsymbol{u}_c$, given the surrounding context words $\mathcal{W}_o = \{w_{o1}, ..., w_{o2m}\}$ and $\bar{\boldsymbol{v}_o} = (\boldsymbol{v}_{o1} + ... + \boldsymbol{v}_{o2m})/(2m)$ for context window size $m$ where the vocabulary index set $\mathcal{V} = \{0, 1, ..., |\mathcal{V}| - 1\}$:

$$P(w_c \mid \mathcal{W}_o) = \frac{\exp(\boldsymbol{u}_c^\top \bar{\boldsymbol{v}}_o)}{\sum_{i \in \mathcal{V}} \exp(\boldsymbol{u}_i^\top \bar{\boldsymbol{v}}_o)} \tag{2.1}$$

For skip-gram, the conditional probability of generating a nearby context word $w_o$ with the context vector $\boldsymbol{u}_o$, given the center word $w_c$ with the center vector $\boldsymbol{v}_c$, can be modeled by a softmax operation on vector dot products:

$$P(w_o \mid w_c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\boldsymbol{u}_i^\top \boldsymbol{v}_c)} \tag{2.2}$$

By predicting a word based on other words that surrounds it with CBOW or predicting other words that surrounds a specific word with skip-gram, word2vec is able to learn and create numerical representations of the meaning of words.

| Word $w_i$ | $\boldsymbol{v}_i[0]$ | $\boldsymbol{v}_i[1]$ | | $\boldsymbol{v}_i[d-1]$ |
|---|---|---|---|---|
| みんな → | 1.406 | 0.417 | ... | 1.771 |
| いる → | 0.055 | -0.683 | ... | 0.865 |
| ⋮ | | | | |
| ごと → | 0.381 | 1.653 | ... | 0.876 |
| | ---------- average ---------- | | | |
| Combined Vector | -0.358 | 0.8304 | ... | 0.629 |

Fig. 2–4: Combining vectors by taking average values of word2vec output

However, just like sentences, lyrics consist of a number of words. The approach proposed in [25], for example, obtains a vector representation of lyrics with word2vec trained with CBOW strategy by combining the vector representations of the words that are present in the lyrics as shown in Fig. 2–4. While there is a logical reasoning behind this approach, there is a concern of how well the final vector can represent the respective lyrics.

According to [29], every lexical word has its meaning, but its social application has implications on our day-to-day communication, as they may have different semantics roles when put together in a context. In other words, the meaning of words may change depending on the presence of other words, which consequently build the context of the sentences, or in this case lyrics. Considering the similarities between lyrics and sentences in terms of them being formed by combining words, it is reasonable to conclude that to process lyrics, a method that can generate more appropriate numerical representation for sentences is preferred.

### 2.3.2 Lyrics Embedding with Sentence-BERT

Recent research has resulted in many powerful language models, such as Bidirectional Encoder Representations from Transformers (BERT) [30], which pre-trains deep bidirectional representations from unlabeled text by conditioning on both sides of a context in all layers. A pre-trained BERT model can be fine-tuned with an additional output layer to create state-of-the-art models for a wide range of tasks, without substantial task-specific architecture modifications.

One big downside of using BERT is the time necessary for semantic textual similarity (STS) task which is the general scope of what we are trying to achieve in this research, as we need to compare lyrics contents to see their similarities which will then be used as a base to estimate musical components. Fortunately, this can be mitigated by using an STS-focused model based on BERT, Sentence-BERT [31], which is a computationally efficient model which, on a GPU, is faster than InferSent [32] by 9% and Universal Sentence Encoder [33] by 55%. It can significantly reduce the time necessary to find the most similar pair of sentences to 5 seconds, from 65 hours when done with BERT in a cluster of 10,000 sentences with hierarchical clustering that requires computation of 50M combinations.

SBERT initializes vector representation of each sentence in its vocabulary by pooling the token embedding with the size of 512×768 from BERT as shown in Fig. 2–5, which enables the generation of a fixed-size representation $\boldsymbol{u}$ for input sentences regardless the length. This pooling is done by either computing the mean or max-over-time of all output vectors of the CLS-token.

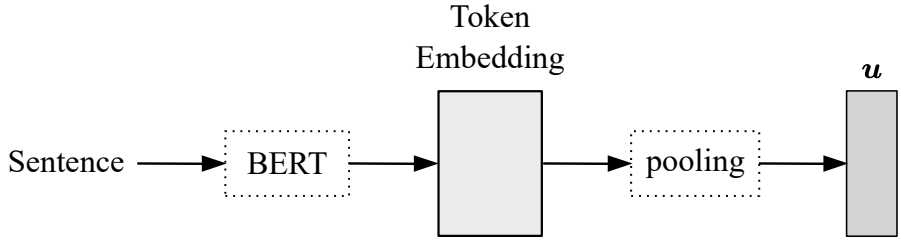


Fig. 2–5: SBERT pooling strategy to create $\boldsymbol{u}$

In training, it processes 2 sentences with their vector representations $\boldsymbol{u}$ and $\boldsymbol{v}$ simultaneously through BERT and a pooling layer on each side and are concatenated with their element-wise difference $|\boldsymbol{u} - \boldsymbol{v}|$ as shown in Fig. 2–6.

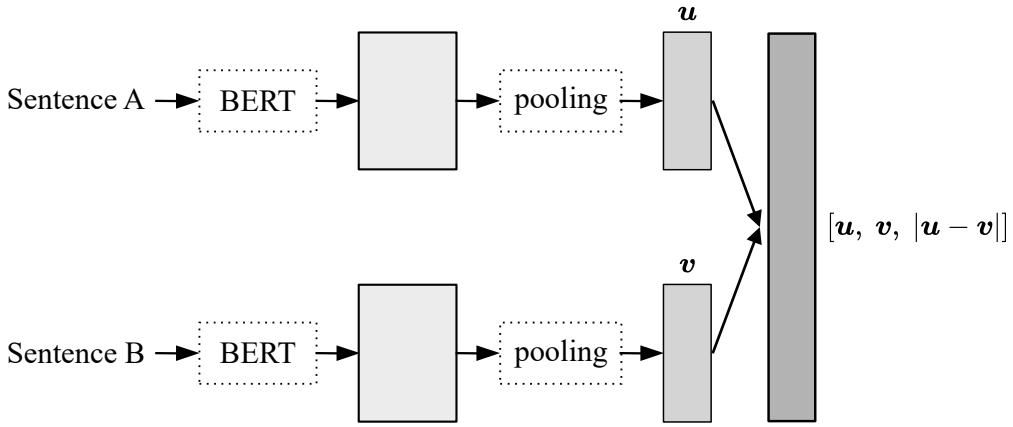


Fig. 2–6: SBERT *twin* network and vectors concatenation

**The Original SBERT Model Training**

The authors combined the Stanford Natural Language Inference (SNLI) [34] and the Multi-Genre NLI (MG-NLI) [35] corpus to create a collection of 1M sentence pairs that are labeled depending how the 2 correlate, 0 for *entailment*, 1 for *neutral*, or 2 for *contradiction*. By feeding the concatenated vectors into a feedforward neural net (FFNN) that outputs $\boldsymbol{o}$ as shown on Fig. 2–7, which can be used to determine the correlation between the 2 sentences.

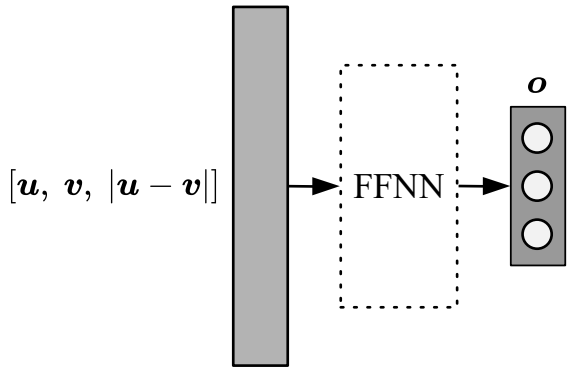


Fig. 2–7: Vector concatenation in SBERT

The calculation of output activation $\boldsymbol{o}$ is done by multiplying the concatenated vectors with the trainable weight $\mathbf{W}_t \in \mathbb{R}^{k \times 3n}$:

$$\boldsymbol{o} = \text{softmax}(\mathbf{W}_t[\boldsymbol{u},\ \boldsymbol{v},\ |\boldsymbol{u}-\boldsymbol{v}|]) \tag{2.3}$$

**The Japanese SBERT Model Training**

There are some strategies that can be used to train an SBERT model depending on the corpus and needs. The Japanese SBERT model that was introduced in [36], for example, is trained with multiple negatives loss as proposed in [37]. This loss takes a batch of size $K$, from which there will be $K$ input $\boldsymbol{x} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_K]$ and other sentences $\boldsymbol{y} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_K]$. The sentence that is the most similar to $\boldsymbol{u}_i$, $\boldsymbol{v}_j$ is located where $i = j$, while the rest of the randomly chosen sentences $\boldsymbol{v}_j$ is treated as a negative candidate for $\boldsymbol{u}_i$ (thus the name multiple negatives).

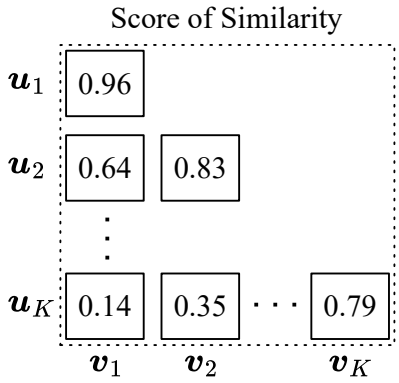


Fig. 2–8: Sentence similarity in model training with multiple negative loss

An example of this is, the sentence A “I want to go.” with the vector representation $\boldsymbol{u}_1$ is paired with the sentence B “I would like to leave.” with the vector representation $\boldsymbol{v}_1$. Other unrelated sentences such as “She loves climbing.” with the vector representation $\boldsymbol{v}_j$ that are randomly sampled are used as the negative candidates of the sentence A, as shown in Fig. 2–8.

As the output of other examples in a training batch of stochastic gradient descent are used as negative output, the $K - 1$ negative examples for each $\boldsymbol{u}_i$ are different at each pass through the data. To minimize the approximated mean negative log probability of the data in a single batch where $\theta$ represents the word embeddings and neural network parameters used to calculate the score of similarity between the 2 sentences $o(\boldsymbol{u}_i, \boldsymbol{v}_j)$:

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{y}, \theta) = -\frac{1}{K} \sum_{i=1}^{K} \left[ o(\boldsymbol{u}_i, \boldsymbol{v}_i) - \log \sum_{j=1}^{K} e^{o(\boldsymbol{u}_i, \boldsymbol{v}_j)} \right] \tag{2.4}$$

By updating the weight $\mathbf{W}_t$ that is used in BERT through training, the SBERT model consequently updates each vector representation $\boldsymbol{u}$ which, with pooling, becomes a semantic embedding rather than just token embedding as BERT originally outputs. In other words, SBERT finetunes the output of BERT by learning the similarities between sentences.

With this strategy, SBERT can be used to convert unknown textual data, specifically lyrics in this case, into their numerical representation which can then be used to train our estimation models. Unlike word2vec that processes text data word by word, SBERT processes them as is, meaning it can convert a sentence, paragraph, or in this case, lyrics, into its semantics embedding as shown in Fig. 2–9. This ensures that the embedding is created not only based on the meaning of words that are present in the lyrics input, but also their semantics in lyrics.

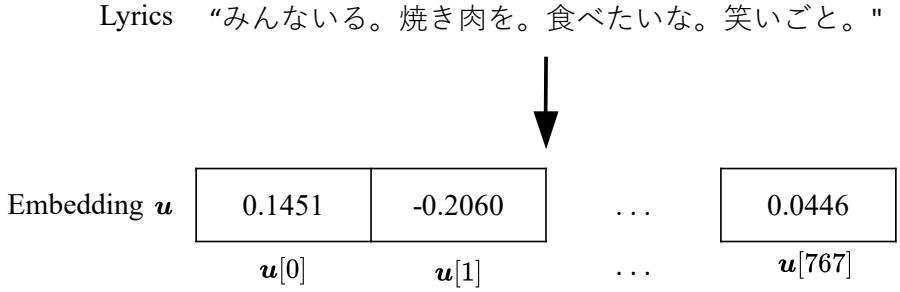


Fig. 2–9: Converting lyrics into embedding with SBERT

## 2.4 Multi Layer Perceptron as a Classifier Model

With the lyrics pre-procession handled by SBERT, it is now possible to develop neural models that are capable of recommending musical components with lyrics input. Considering that SBERT itself is already a complex language model, we decided to balance our system by using a simple multi layer perceptron (MLP) model [38] as a classifier. This is done to avoid overfitting that happens due to using overly complex architecture.

### 2.4.1 How MLP Works

A perceptron, as proposed in [38], is an algorithm for supervised learning of binary classifiers which imitates a brain neuron in terms of how it learns. It consists of 3 components: a function, input, and output.

A multi-layer perceptron is a feedforward artificial neural network with fully connected class. It produces an output based on an input and its function, and it learns from labeled data by a process of guessing the label of data with hidden labels and correcting its guess when it does not match the correct label. This correction process is repeated in a number of predetermined epoch.

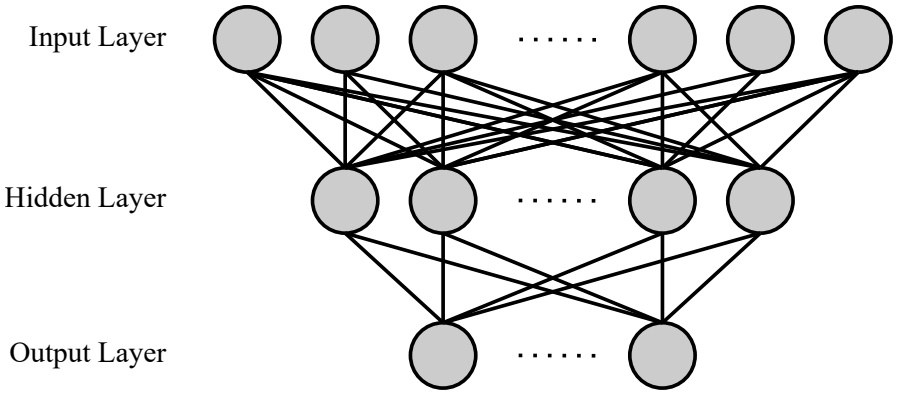


Fig. 2–10: A 3-layer MLP with big input and small output layers

Using this concept, it is possible to create an architecture that takes in a certain dimension of numerical value array and resizes it to a different size through the hidden layer(s) which usually ends up with an output of a smaller dimension in comparison to the input as shown in Fig. 2–10. This makes it possible to convert semantics embedding to its perceived array of classification, subsequently resulting in an estimate of which class it belongs.

### 2.4.2 Accuracy as the Training Metrics

To train an MLP, it is necessary to decide on the training metrics to monitor and measure the performance of a model. Considering that musical components are estimated based on lyrics and a list of known possible output, this study case falls into classification task. For this purpose, the accuracy $A$ is used, as it calculates the number of correct prediction $a$ by the model in relation to the total number of prediction made $n$, in scale of 0 to 1, calculated as follows:

$$A = \frac{1}{n} \sum_{i=1}^{n} a_i \tag{2.5}$$

### 2.4.3 The Effects of Data Imbalance in Machine Learning

As previously discussed in 1.2.2, using Orpheus as a source of training data risks in data imbalance. Simply excluding options with low number of samples, still does not guarantee that the remaining options have similar number of samples. This imbalance can cause a problem in model training when not mitigated properly, as the less number of samples an option has, the more difficult for an AI model to learn it. Despite using the data extracted from Orpheus, this problem was not addressed in [25], which most likely was the cause of their model inability to learn from the dataset properly and the low accuracy.

In machine learning, it is common to use cross-entropy (CE) as a loss function. It is a measure from the field of information theory, building upon entropy. It generally calculate the difference between two probability distributions and is extendable to be used for categorizing input into several classes, which fits the use case of this research. Cross-entropy is defined as follows, where $p_i$ is the softmax probability of the $i^{th}$ class:

$$L_{CE} = -\sum_{i=1}^{n} \log(p_i) \tag{2.6}$$

However, this categorical CE treats all classes equally, regardless the number of samples. This causes the trained model to overfit on classes with low number of samples, as they lack in variety of data. To properly train a model with an imbalanced dataset, a different strategy needs to be applied.

### 2.4.4 Applying Focal Loss to Reduce Overfitting

An imbalanced data requires more attention on classes with low number of samples in order to reduce overfitting. According to [39], applying a focal factor $\gamma = 2$ allows a model to down-weight classes with higher number of samples and focus more on those with lower number of samples. The focal loss (FL) function for multiclass classification with categorical CE is:

$$L_{FCE} = -\sum_{i=1}^{n}(1 - p_i)^{\gamma} \log(p_i) \tag{2.7}$$

## 2.5 Evaluating the Model Objectively and Subjectively

While lyrics-based musical components estimation can be automated with machine learning, there is still a necessity of evaluating the proposed approach, as we need to make sure that an approach is reliable or not. After all, there is no point of using a complex approach if the resulting estimations are not better than baseless outputs in terms of how they correlate to the input.

### 2.5.1 Objective Evaluation with ROC

In [23], Receiver Operating Characteristic (ROC) curve was used to evaluate the performance of their model. ROC is a graphical plot of the True Positive Rate (TPR), which is the rate of test results classified as true and match the ground truth, against the False Positive Rate (FPR), which is the rate of test results classified as true but do not match the ground truth, as shown in Fig. 2–11.

This plot illustrates the diagnostic ability of a classifier system, which makes it suitable for this research. As proposed in [40], the score in ROC evaluation is based on the area under curve (AUC), which is calculated using Trapezoidal Rule Numerical Integration method as follows:

$$\text{ROC AUC} = \frac{(\text{FPR}_{i+1}\text{–FPR}_i)(\text{TPR}_i + \text{TPR}_{i+1})}{2} \tag{2.8}$$

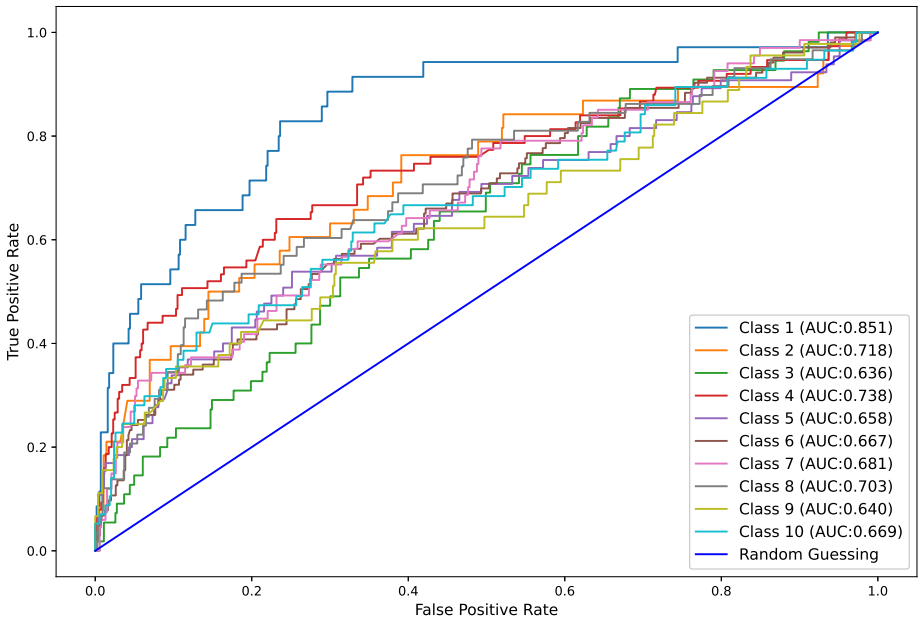


Fig. 2–11: An ROC graph of a multi-class classification system

This approach, however, may not be the best for our case if we also consider the imbalance in data, as it treats all classes equally regardless the number of samples, which may be better represented by using different metrics.

### 2.5.2 Objective Evaluation with F1 Score

Unlike ROC AUC, F1 score considers the total number of samples of all class instead of that of each individual class, and for this reason, objective evaluation with F1 score may be a more suitable approach in this research. It is calculated from the precision $p = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and recall $r = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ of the test, where TP is the number of True Positives, FP is the number of False Positives, and FN is the number of False Negatives in the rest results:

$$\mathrm{F1} = 2\frac{pr}{p+r} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{2.9}$$

### 2.5.3 Subjective Evaluation through Surveys

In addition to these 2 objective evaluation methods, it is also necessary to evaluate the performance of the proposed model subjectively to see whether the output is of a higher quality from human perspective. A simple way to do this is comparing the degree of approval of several subjects in a survey, as proposed in [41].

However, depending on the relevant knowledge of respondents on the topic, there is a possibility of bias caused by the difference in understanding, confidence, and other factors. For this reason, weighting their responses might be necessary during the evaluation in order to reduce the aforementioned bias.

# Chapter 3

# Utilizing SBERT and Focal Loss Function

## 3.1 Problem Definition and Proposed Solution

In chapter 1, we have discussed the importance of automating the musical component setup in complex AI-based music composition system and that it can potentially be achieved with machine learning. This potential is further discussed in 2, followed by an introduction to prior research that can be used as a base to work and other related works that support our arguments and hold the keys to help us realize the goal of this research.

In this research, we propose an MLP model that recommends 5 musical components with lyrics input by using SBERT for pre-processing and focal loss function during model training. However, before deciding on how to utilize SBERT and focal loss function, we first need to dissect the composition data extracted from Orpheus [7] to figure out how to best handle them.

## 3.2 The Top 10 Lyrics-Musical Component Datasets

Orpheus provides various composition setup data which includes but not limited to lyrics and the 5 musical components we use as research subjects. For this research, we filtered out irrelevant data and sampled the composition per part instead of piece to compile higher number of samples, resulting in over 24,000 samples that can be used as training data. A sample of filtered setup data can be seen on table 3–1, which includes lyrics and the label of musical components options used in the setup done by the original composer.

By combining different options with their lyrics, users can generate unique compositions. However, it is not mandatory to specify all musical components in this setup. As a result, some lyrics in the data extracted from Orpheus may not have one or more of these musical components, resulting in different number of samples for each pair of lyrics and musical component.

Table 3–1: A filtered setup data sample of a published composition in Orpheus

| Lyrics | Chord Progression | Rhythm Pattern | Instrument | Tempo | Drum Pattern |
|---|---|---|---|---|---|
| からまつの林を過ぎて、<br>からまつをしみじみと見き。<br>からまつはさびしかりけり。<br>たびゆくはさびしかりけり。 | Pachelbel-Kanon | sync-auf-3-8sf | 48 | 100 | perc-hirata-rocknroll2 |

As it has been discussed in 1.2.2, it is necessary to exclude data with low number of samples to avoid overfitting. This is done by including only the 10 options for each musical component that are popular amongst the users. These are the setups that are often chosen by users, which consequently means that they have the most number of samples, sufficient for training.

### 3.2.1 The Top 10 Chord Progression Dataset

In Orpheus, each chord progression consists of a sequence of 16 chords. Pattern O, the class with the most number of samples, for example, as shown on 3–2, is a sequence of the following chords: CM → Em7 → Am7 → Am7 → FM7 → Bm7 → Dm7 → G7 → CM → Em7 → Am7 → Am7 → FM7 → Dm7 → G7 → CM. Refer to A–1 for a list of the chord sequence for each option in the top 10 dataset.

All chord sequences with audio samples are listed in [42] for reference. Compared to other musical components, there are only 5,980 pairs of lyrics and chord progression, with 1,113 samples at most and 386 at the lowest. This is most likely caused by the fact that there are over 1,500 chord progression options to choose from, which might have overwhelmed the users.

Table 3–2: Top 10 chord progressions in published Orpheus data

| Option Label | #Samples |
|---|---|
| pattern O | 1,113 |
| pattern FF | 939 |
| pattern Q | 596 |
| pattern P | 562 |
| pattern H | 560 |
| pattern E | 535 |
| pattern W | 503 |
| pattern R | 397 |
| Pachelbel Kanon Ending | 389 |
| User Harmony zkrxx7 | 386 |

### 3.2.2 The Top 10 Rhythm Pattern Dataset

In Orpheus, there are simple and complex rhythm patterns which are based on famous songs, such as Love Machine. Unlike chord progression, there are twice more lyrics data paired with rhythm pattern, resulting in a total of 11,674 samples, with at least 562 samples for each option up to 4,397 at the highest as seen on table 3–3. This significant difference might have been caused by the equally significant difference in number of options available for rhythm pattern in Orpheus, as there are only 64 options of rhythm patterns available to choose from during composition setup. Refer to appendix B for musical score of each option in the top 10 dataset or [43] for sample compilations of rhythm patterns available on Orpheus.

Table 3–3: Top 10 rhythm patterns in published Orpheus data

| Option Label | #Samples |
|---|---|
| pattern A32 | 4,397 |
| syncopee 01ss | 1,310 |
| love machine | 1,048 |
| school 1ss | 878 |
| Glass no shounen Kinkikids | 826 |
| pattern L32 | 717 |
| pattern J32 | 664 |
| sync-auf-3-16sf | 657 |
| chijo no hoshi 1ss | 615 |
| march 110815ss | 562 |

### 3.2.3 The Top 10 Instrument Dataset

Instruments in Orpheus are labeled with number, so for readability, we have included label translation in addition to their original tag number from Orpheus in table 3–4. In terms of the total number of samples, this dataset is similar to that of rhythm pattern, with 10,535 samples despite the fact that there are around twice as many options for instrument compared to rhythm pattern with a total of 127 variations. This is most likely caused by the familiarity with how a musical instrument sounds that most people have a general knowledge of. This dataset is not as imbalanced, as there is a smaller gap between the option with the highest number of samples, which only reached 2,446, and that with the lowest number of samples of 584. A full list of the options with samples can be seen in [44].

Table 3–4: Top 10 instruments in published Orpheus data

| Option Label | #Samples |
|---|---|
| 25. Acoustic Guitar (Steel Strings) | 2,436 |
| 0. Acoustic Piano | 1,524 |
| 29. Overdrive Guitar | 1,292 |
| 2. Electric Grand Piano | 1,071 |
| 24. Acoustic Guitar (Nylon Strings) | 817 |
| 27. Clean Guitar | 815 |
| 48. String Ensemble 1 | 709 |
| 30. Distortion Guitar | 679 |
| 1. Bright Piano | 608 |
| -1. No Instrument | 584 |

### 3.2.4 The Top 10 Tempo Dataset

In Orpheus, the tempo composition can be set from 54 to 270 beats per minute (BPM). It is also possible to set it manually, making it possible to set it with unreasonably high or low BPM. We decided to exclude such data and instead of treating each of those as one separate option, we divided them into groups in increments of ten before taking the top 10 off them as shown on 3–5, following the approach used in [23] that they used to group the release year of songs. It is likely that since it is easier to set, we managed to compile 22,594 lyrics samples paired with tempo data, with at least 389 samples each option, up to 4,700 at most.

Table 3–5: Top 10 tempi in published Orpheus data

| Option Label | #Samples |
|---|---|
| 111 - 120 | 4,700 |
| 121 - 130 | 4,362 |
| 131 - 140 | 3,464 |
| 81 - 90 | 2,393 |
| 91 - 100 | 2,192 |
| 151 - 160 | 1,881 |
| 101 - 110 | 1,863 |
| 181 - 190 | 920 |
| 141 - 150 | 430 |
| 161 - 170 | 389 |

### 3.2.5 The Top 10 Drum Pattern Dataset

The options of drum patterns in Orpheus are labeled according to their genre and number of variations. For this reason, we grouped these drum patterns based on their genre before creating the top 10 list as seen on 3–6. Having a low number in options, it is likely that it was seen to be easier to set by the users, resulting in total number of samples that is slightly higher than tempo. A total of 22,862 samples are available, with smaller gap between the first and last options in the top 10 list with 4,403 and 732 samples respectively. Audio samples of the drum patterns are compiled in a composition available in [45].

Table 3–6: Top 10 drum patterns in published Orpheus data

| Option Label | #Samples |
|---|---|
| Empty | 4,403 |
| Amero | 4,083 |
| Rock | 3,779 |
| Ballad | 2,542 |
| Rock&Roll | 1,942 |
| Funk | 1,861 |
| 8-Beat | 1,837 |
| Fusion | 947 |
| Jazz | 736 |
| Tom | 732 |

## 3.3 SBERT Model Pre-trained with Japanese Corpus

For an SBERT model to be able to generate an embedding, it is necessary to do pre-training. This is done by feeding the model with unlabeled text for it to base on. BERT is equipped with a big library of pre-trained model and we can choose which to use depending on the language of the training dataset. Considering the language of the dataset used in our research, we decided use the SBERT model pre-trained with Japanese corpus.

This pre-trained SBERT model shares the same architecture with the original BERT base model; 12 layers, 768 dimensions of hidden states, and 12 attention heads. It is pre-trained on Japanese Wikipedia as of September 1, 2019, with a total size of 2.6GB training data, consisting of approximately 17M sentences. The pre-training data was tokenized using MeCab morphological parser with the IPA dictionary as proposed in [46] and then split into subwords using the WordPiece algorithm introduced in [47] resulting in a vocabulary size of 32,000.

## 3.4 Setting Up an MLP as the Proposed Classifier Model

Having decided on the pre-trained SBERT model, we now need to compile a neural model to take SBERT output as training data. As mentioned in 2.4, the use of SBERT model makes it possible to use simple architecture for training. This simplicity will also allow us to focus on how to train the model more, giving insight for future research which can then take on more complex architecture.

### 3.4.1 MLP Model Architecture

We keep our MLP architecture simple by limiting it to 3 layers, input, hidden, and output as shown in Fig. 3–1. The function of these layers are as follows:

#### The Input Layer X

The input layer X takes in embedding converted from lyrics with SBERT and size it down to be forwarded to the hidden layer. The size of the input here reflects the size of the embedding itself and this differs according to what SBERT model is being used. As mentioned in 3.3, the size of input will be 768, matching the output dimension of the Japanese SBERT model.

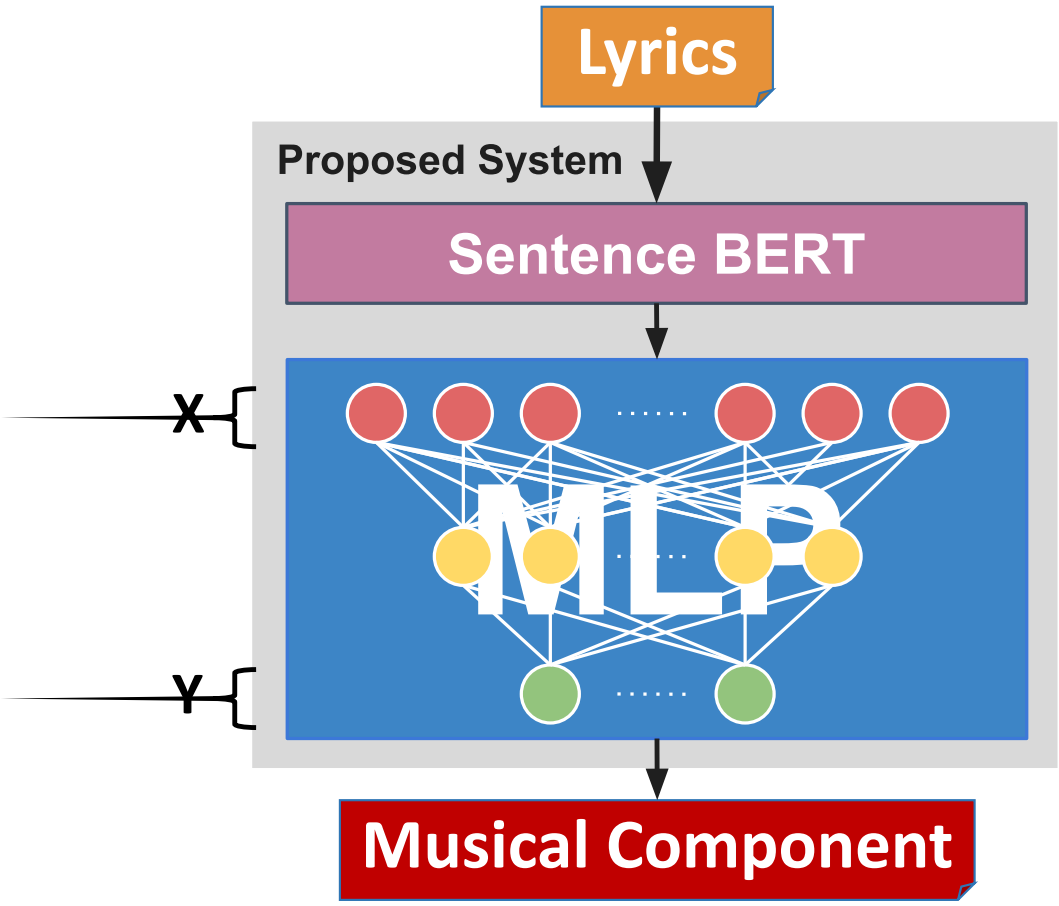


Fig. 3–1: The architecture of the proposed classifier model

### The Output Layer Y

As previously discussed in 3.2, options with insufficient number of samples are excluded by limiting the dataset to the top 10 of each musical components. This affects the size of the output layer Y, which as a result is set to 10 to match the number of possible options as the output.

### The Hidden Layer

In MLP, the hidden layer is located in between the input and output layers. It functions as a bridge between them. For complex data processing, multiple hidden layers are necessary. However, as we are trying to keep the classifier model simple, we keep the number of layers low by only setting up one hidden layer. The dimension of this layer is set to 77, a tenth of the that of the input layer.

### 3.4.2 Accuracy as Training Metrics and Focal Loss Function

As mentioned in 2.4.2, it is necessary to choose proper metrics for the model training based on the study case. In this research we train our proposed model with accuracy as training metrics considering that estimating musical components based on lyrics falls into the classification task.

To minimize the overfitting caused by imbalance in data, we use the focal loss (FL) function extended for multi-class case as mentioned in 2.4.4. With this approach, we hope to achieve better performance in terms of music relevancy to lyrics compared to the model that recommends chord progression with lyrics input proposed in [25], in which word2vec was used for pre-processing and no solution was offered to solve the overfitting problem caused by using datasets with imbalanced number of samples for training.

# Chapter 4

# Experiments with the Proposed Model

To find the best model setup that can solve this estimation problem, we experimented on the 5 datasets of the musical components, each with a combination of 1 of the 3 pre-processing approaches and 2 different loss functions during training. Other attempts done in this research that did not improve the model performance but are worth mentioning will be explained in 4.3.

## 4.1 Lyrics Pre-processing and Training Loss Function

We first combined 3 lyrics pre-processing methods and 2 loss functions during training, resulting in a total of 6 variations as shown in 4–1. For the training data, we used chord progression dataset previously explained in 3.2.1 considering its small number of samples and importance in composition process as stated in [11]. The lyrics from said dataset are pre-processed with a rebuilt of the word2vec (W) model proposed in [25] and 2 SBERT models pre-trained with the Japanese (J) corpus as mentioned in 3.3 and multi-language (M) corpus that does not include Japanese, but Chinese, which also utilizes the kanji system.

This comparison is done to see if SBERT can still pre-process the lyrics properly by learning from a similar language. The word2vec (W) model is pre-trained with the same, although more recent corpus used by the Japanese SBERT model which uses Japanese Wikipedia as of June 20, 2023. This difference is due to the unavailability of the corpus from 2019 used in Japanese SBERT pre-training.

Table 4–1: Models abbreviation and their differences

| Abbreviation | Pre-training Corpus | Pre-processing | Loss Function |
|---|---|---|---|
| WC | Japanese Wikipedia 2023 | Word2Vec (W) | Categorical CE (C) |
| WF | | | Categorical Focal CE (F) |
| MC | Multi-language (M) | SBERT | Categorical CE (C) |
| MF | | | Categorical Focal CE (F) |
| JC | Japanese Wikipedia 2019 (J) | | Categorical CE (C) |
| JF | | | Categorical Focal CE (F) |

During training of 1,000 epochs, we compare 2 different loss functions, categorical CE (C) and categorical focal CE (F) as mentioned in 3.4.2 to see if applying focal factor can help reduce the overfitting caused by data imbalance.

## 4.2 Final Accuracy and Overfitting Observation

In our experiments, we use model accuracy in picking the same option as what the actual user had chosen as training metrics. We will first display the training results of the 6 variants on the chord progression dataset, followed by those of the best setup and the previous model WC on all musical components.

### 4.2.1 The Effects of Pre-processing Method Modification

We first observed the improvement in model accuracy to see how much changing the pre-processing method will improve how well the models can replicate the ground truth. For this purpose, we built 3 models with different pre-processing methods trained with cross-entropy loss function (C).

The first model WC is a rebuilt of the word2vec model as proposed in [25] that is pre-trained with 2023 Japanese corpus. By training this model with cross-entropy loss function, as seen on Fig. 4–1, the final accuracy values of WC are quite low, with training around 37.3% and 21.4% for validation.

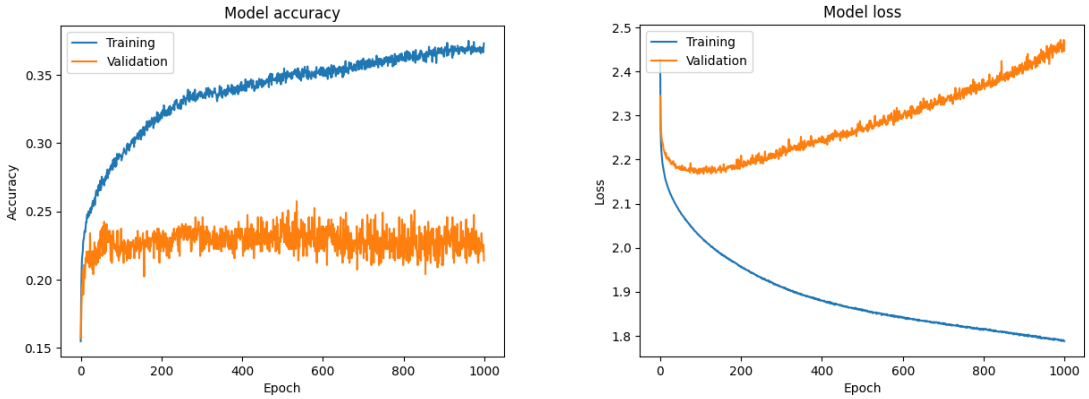


Fig. 4–1: WC accuracy and loss over time during training

Next, we will see if changing the pre-processing model to the base multi-language SBERT will result in higher accuracy values by training MC. With the same setup as WC, MC achieved higher final accuracy values. In training, it reached 50.6%, significantly higher than WC. However, validation accuracy only increased to 26% as shown in Fig. 4–2. This shows that MC still does not do well on unseen data.

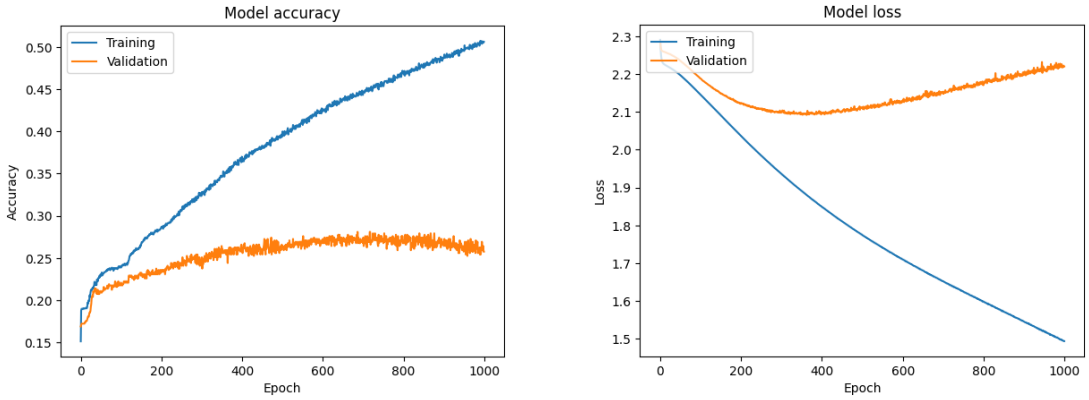


Fig. 4–2: MC accuracy and loss over time during training

Lastly, we will see if changing the pre-training to specifically Japanese corpus will further improve the accuracy values of the model by training JC. Looking at Fig. 4–3, the training accuracy has increased even more significantly up to 96%, while validation accuracy increased to 31.2%. We can see that changing the pre-processing method as proposed in this research successfully resulted in higher model accuracy values, both on seen and unseen data, as shown in 4–2.

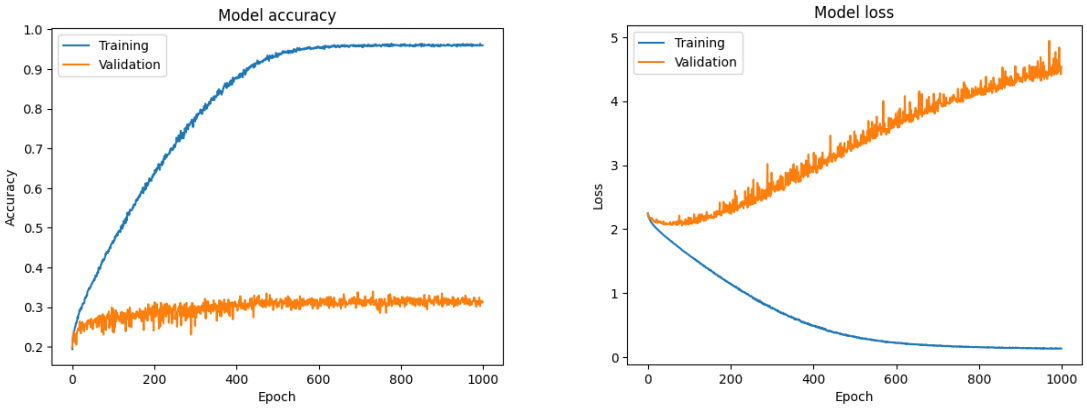


Fig. 4–3: JC accuracy and loss over time during training

However, based on the loss graphs in Fig. 4–1, 4–2, and 4–3, overfitting can be seen to start around 100, 400, and 100 epochs respectively. It can also be seen that the overfitting is the worst on JF. This might have been caused by using unnecessarily high number of epochs, but we decided to let the models keep learning for reasons that will be discussed more in 4.3. The next experiment will cover our attempt to minimize it by changing the loss function in the model training.

Table 4–2: Final accuracy of the 3 models with different pre-processing methods

| **Model** | **T. Acc.(%)↑** | **V. Acc.(%)↑** |
|---|---|---|
| WC | 37.3 | 21.4 |
| MC | 50.6 | 26.0 |
| JC | **96.0** | **31.2** |

### 4.2.2 The Effects of Loss Function Modification

Despite having truncated the datasets to exclude data with low number of samples by creating a top 10 list for each musical components, there is still a noticeable gap between the first and last options in the list. This imbalance makes the model training prone to overfitting as mentioned in 2.4.4, which led us to conducting this experiment. We modified the 3 models in the previous experiment by changing the loss function used in training and observed the effects on the overfitting.

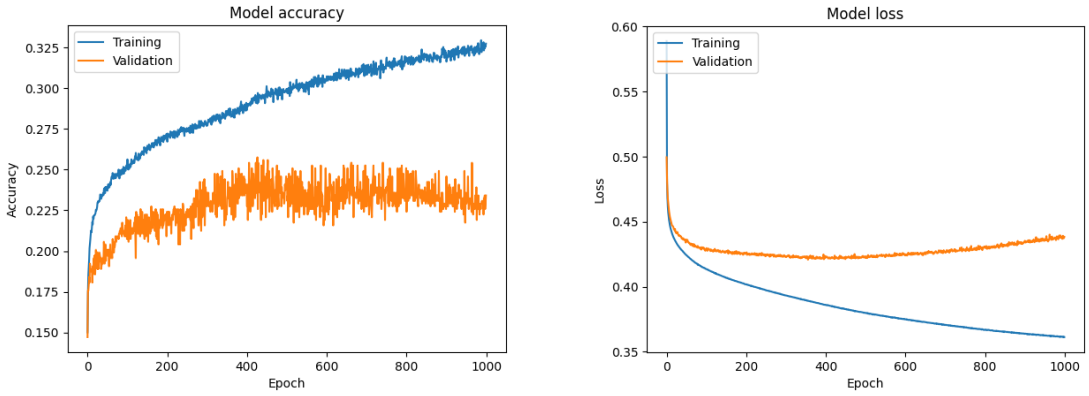


Fig. 4–4: WF accuracy and loss over time during training

By changing the loss function on the word2vec model into focal cross-entropy to build WF, it managed to achieve noticeably less overfitting, based on the loss graph shown in Fig. 4–4. Although the training accuracy reached 32.7% which is slightly lower than WC, it is higher slightly higher on validation with 23.4%.

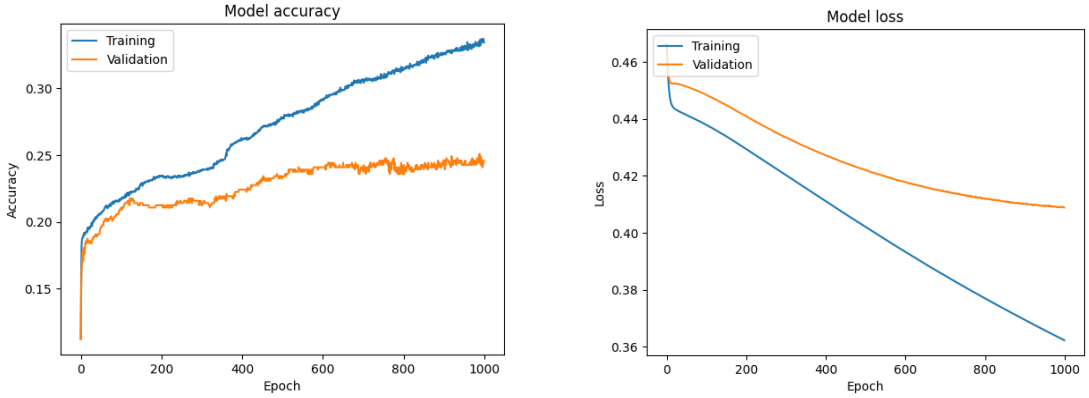


Fig. 4–5: MF accuracy and loss over time during training

This improvement in terms of overfitting, can also be seen on the multi-language SBERT model trained with focal loss function MF as shown in Fig. 4–5. The final accuracy values, however, dropped on both training and validation, with 33.4% and 24.6% respectively. This is most likely caused by the exclusion of Japanese data in the corpus. Although the multi-language still includes similar language, which is Chinese, it seems that it is still not enough to properly learn the data.

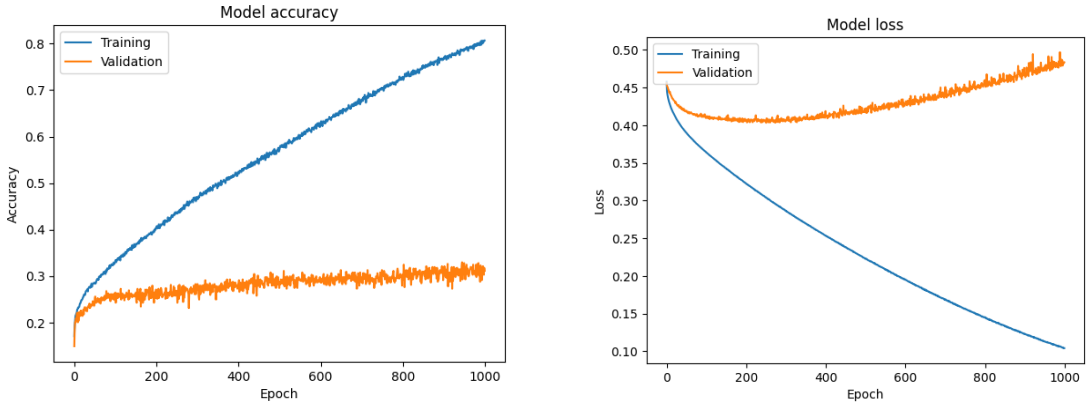


Fig. 4–6: JF accuracy and loss over time during training

By using a more appropriate pre-training corpus for the pre-processing and focal loss function in model training, the Japanese SBERT model JF achieved comparable overfitting, although not as minimized as WF. As it achieved training accuracy value of 80.7% and maintained the validation accuracy value of 31.1%, we are hopeful that this model can yield promising results in the evaluation.

Table 4–3: Final accuracy of the 3 models trained with focal loss function

| **Model** | **T. Acc.(%)↑** | **V. Acc.(%)↑** |
|---|---|---|
| WF | 32.7 | 23.4 |
| MF | 33.4 | 24.6 |
| JF | **80.7** | **31.1** |

In summary, as shown in table 4–2 and 4–3, we can see that the Japanese SBERT models JF performs the best, with competitively minimum overfitting. However, we still need to ensure that this is consistent on other musical components, which brings us to the next experiment.

### 4.2.3 Model Training with the 5 Musical Components Dataset

After experimenting with the pre-processing method and loss function on the chord progression dataset, we compared WC, a rebuilt of the model proposed in [25], with our best model JF on the remaining 4 musical component datasets.

Table 4–4: Final accuracy of WC and JF on all 5 musical components

| **Dataset** | **Model** | **T. Acc.(%)↑** | **V. Acc.(%)↑** |
|---|---|---|---|
| Chord Progression | WC | 37.3 | 21.4 |
| | JF | **80.7** | **31.1** |
| Rhythm Pattern | WC | 41.4 | 36.9 |
| | JF | **79.0** | **38.7** |
| Instrument | WC | 31.7 | 28.4 |
| | JF | **78.0** | **32.7** |
| Tempo | WC | 27.8 | 23.9 |
| | JF | **70.0** | **28.4** |
| Drum Pattern | WC | 25.9 | 21.9 |
| | JF | **64.7** | **26.4** |
| Average ± Deviation | WC | $32.8 \pm 6.5$ | $26.5 \pm 6.4$ |
| | JF | $\mathbf{74.5 \pm 6.8}$ | $\mathbf{31.5 \pm 4.7}$ |

As shown in 4–4, JF consistently has higher accuracy compared to WC, meaning that our model was able to perform better in training. Judging from these results alone, we can see that SBERT pre-processing topped with categorical focal CE is better than word2vec pre-processing with categorical CE on training and validation data. To see whether these results are also consistent on test data, evaluations which compare WC and JF are necessary, which will be covered in 5.

## 4.3 Ablation Studies

While it has been reduced by applying our approach, we can still see overfitting happening in the proposed model JF. As previously mentioned in 4.2.1, it is possible that this overfitting happens due to the overly long epoch. However, cutting down the epoch risks the models having higher final loss and lower accuracy. Unfortunately, by using early stop to stop the training before the overfitting happens, the models end up having higher final loss and lower accuracy as we expected, which is why we decided to keep the epoch at 1,000. We have also considered the opposite possibility of the models having incomplete learning due to insufficient epoch. To make sure that it is not the case, we doubled our epoch to 2,000, which also resulted in higher final loss and lower accuracy.

Besides overfitting, the decrease in loss and the increase in accuracy are not as high as we have hoped to see. In order to ensure that we have chosen the best setup, we attempted several approaches, including: experimenting with dropout layer of 0.2 and 0.8, changing the $\alpha$ of the focal loss function according to the number of samples in each class, using sample weights during training, and changing the optimizer to adam instead of Stochastic Gradient Descent (SDG). None of these attempts yield better results compared to JF, which proves that the current setup of our proposed model JF is the best that we can find in terms of having the minimum overfitting, lowest loss, and highest accuracy.

# Chapter 5

# Model Evaluation and Discussion

We evaluate our proposed model JF objectively and subjectively. The objective evaluation is done to see how much better it is in replicating Orpheus user selection in comparison with the model WC that was rebuilt based on [25] with ROC and F1 score as previously mentioned in 2.5.

In addition to that, to see if our model JF can recommend musical components to be used in automatic composition with a high satisfaction rate, surveys that directly compares audio samples were conducted as a mean of subjective evaluation. These are all done separately for each musical component test data and summarized for the conclusion.

## 5.1 Objective Evaluation with ROC

To see how well our models are able to replicate musical components selection by Orpheus user, we evaluated our models by observing the Area Under the Curve (AUC) of ROC of the top 10 datasets of the 5 musical components.

### 5.1.1 ROC Evaluation on Chord Progression Options

Looking at 5–1, it is very clear that JF is can estimate chord progression better than WC with an average roc score of 69.6%, which is 5.4% higher than WC, although with slightly higher deviation of 0.9%. This improvement is not consistent however, as we can see a decrease in pattern E and R.

Table 5–1: ROC AUC score in top 10 chord progression options

| Option Label | WC ROC (%) | JF ROC (%) |
|---|---|---|
| pattern O | 60.1 | **65.8** |
| pattern FF | 59.3 | **63.6** |
| pattern Q | 75.0 | **85.1** |
| pattern P | 69.2 | **71.8** |
| pattern H | 61.6 | **68.1** |
| pattern E | **67.4** | 66.7 |
| pattern W | 67.1 | **73.8** |
| pattern R | **65.4** | 64.0 |
| Pachelbel Kanon Ending | 59.6 | **70.3** |
| User Harmony zkrxx7 | 57.7 | **66.9** |
| Average ± Deviation | $64.2 \pm 5.5$ | height±6.4 |

### 5.1.2 ROC Evaluation on Rhythm Pattern Options

For rhythm pattern, we can see that all options are better estimated by JF regardless of how many samples they have, with an average increase of 7.6% from WC, which results in an average score of 67.4% on JF. As seen on 5–2, pattern A32 and love machine which are in the top 3 have the least increase of 3.1%, followed by march 110815ss which is the option with the lowest number of samples. Interestingly, syncopee 01ss, which is the second highest in terms of number of samples, has the highest increase, followed equally by pattern J32, sync-auf-3-16sf, and chijo no hoshi 1ss. This noticeable difference can be seen from the increase in deviation of 1.5% from WC to JF.

Table 5–2: ROC AUC score in top 10 rhythm pattern options

| Option Label | WC ROC (%) | JF ROC (%) |
|---|---|---|
| pattern A32 | 59.8 | **62.9** |
| syncopee 01ss | 61.2 | **73.5** |
| love machine | 59.8 | **63.5** |
| school 1ss | 62.0 | **67.7** |
| Glass no shounen Kinkikids | 57.0 | **64.4** |
| pattern L32 | 58.3 | **65.5** |
| pattern J32 | 56.1 | **67.3** |
| sync-auf-3-16sf | 63.2 | **73.1** |
| chijo no hoshi 1ss | 59.1 | **69.7** |
| march 110815ss | 60.9 | **66.2** |
| Average ± Deviation | $59.7 \pm 2.2$ | $\mathbf{67.4 \pm 3.7}$ |

### 5.1.3 ROC Evaluation on Instrument Options

The third of the top 10 instruments, Overdrive Guitar, leads with 10.6% increase in ROC AUC as seen on 5–3, followed closely by Acoustic Guitar (Steel) which has the most number of samples and Electric Grand Piano which was the fourth in the top 10 list, with 9.2% and 8.9% increase. While the rest varies in ROC AUC increase value, a decrease of 2.6% can be seen on No Instrument. Interestingly, with an ROC score increase of 5.3%, the deviation also drops for this test data.

Table 5–3: ROC AUC score in top 10 instrument options

| Option Label | WC ROC (%) | JF ROC (%) |
|---|---|---|
| 25. Acoustic Guitar (Steel Strings) | 54.7 | **63.9** |
| 0. Acoustic Piano | 58.8 | **67.2** |
| 29. Overdrive Guitar | 56.6 | **67.2** |
| 2. Electric Grand Piano | 58.2 | **67.1** |
| 24. Acoustic Guitar (Nylon Strings) | 58.7 | **64.2** |
| 27. Clean Guitar | 57.2 | **58.6** |
| 48. String Ensemble 1 | 56.7 | **62.8** |
| 30. Distortion Guitar | 56.8 | **58.3** |
| 1. Bright Piano | 64.2 | **68.6** |
| -1. No Instrument | **79.7** | 77.1 |
| Average ± Deviation | $60.2 \pm 7.3$ | $\mathbf{65.5 \pm 5.4}$ |

### 5.1.4 ROC Evaluation on Tempo Options

The table 5–4 shows that the tempo option 101-110 improved significantly, with an increase of 14.8% as shown in 5–4, followed by 81-90 with 10.8%. The remaining are slightly lower, while the least increase, interestingly happens on three consecutive ranges of tempo, 121-130, 131-140, 141-150, with a 3.8-4.9% increase, which is closely followed by 91-100 with a 5% increase. Overall, the improvement can consistently be seen on this test data, with an average increase of 7.6% from WC, resulting in 66.2% on JF, although with a slightly higher deviation.

Table 5–4: ROC AUC score in top 10 tempo options

| Option Label | WC ROC (%) | JF ROC (%) |
|---|---|---|
| 111-120 | 58.3 | **67.0** |
| 91-100 | 58.7 | **63.7** |
| 121-130 | 56.6 | **61.5** |
| 131-140 | 57.1 | **61.1** |
| 81-90 | 56.0 | **66.8** |
| 151-160 | 62.1 | **68.8** |
| 101-110 | 52.6 | **67.4** |
| 181-190 | 65.1 | **73.4** |
| 141-150 | 61.0 | **64.8** |
| 161-170 | 59.1 | **67.8** |
| Average ± Deviation | $58.7 \pm 3.5$ | $\mathbf{66.2 \pm 3.6}$ |

### 5.1.5 ROC Evaluation on Drum Pattern Options

As shown in 5–5, the drum pattern option Amero and Fusion are leading in the previous model WC, and the proposed model JF was able to help the remaining options catch up with them. The highest improvement happens on 8-Beat, with an increase of 11.7% followed by Funk and Ballad with 8 and 8.3% increase. The remaining options have less significant increases, ranging from 3.6-5.1%, except for jazz with least increase of 1.4%. In average, ROC score increase of 5.5% can be seen on this test data with similar deviation, resulting in 61.4% on JF.

Table 5–5: ROC AUC score in top 10 drum pattern options

| Option Label | WC ROC (%) | JF ROC (%) |
|---|---|---|
| Empty | 55.6 | **60.7** |
| Amero | 59.6 | **63.2** |
| Rock | 55.6 | **59.8** |
| Ballad | 54.9 | **63.2** |
| Rock&Roll | 52.5 | **56.4** |
| Funk | 57.2 | **65.2** |
| Fusion | 59.4 | **63.4** |
| 8-Beat | 53.0 | **64.7** |
| Jazz | 57.6 | **59.0** |
| Tom | 54.6 | **59.0** |
| Average ± Deviation | $56.0 \pm 2.4$ | $\mathbf{61.4 \pm 2.9}$ |

### 5.1.6 Overall Evaluation Result with ROC AUC

Overall, we see improvement in all 5 musical components as shown in 5–6 by changing the pre-processing and loss function of our models. The highest increase can be seen on rhythm pattern, closely followed by tempi. The remaining 3, drum patterns, chord progression, and instrument have a similar increase in ROC AUC. In terms of having the highest ROC AUC at JF, chord progressions is leading, followed by rhythm patterns, tempi, instruments, and lastly, drum patterns. With this result, it is safe to conclude that our proposed model JF is capable of replicating human preference in selecting musical components based on lyrics with higher accuracy in comparison to the previous model WC.

Table 5–6: ROC AUC score in all 5 musical components

| Musical Components | WC ROC (%) | JF ROC (%) |
|---|---|---|
| Chord Progressions | 64.2 | **69.6** |
| Rhythm Patterns | 59.8 | **67.4** |
| Instruments | 60.2 | **65.5** |
| Tempi | 58.7 | **66.2** |
| Drum Patterns | 56.0 | **61.5** |
| Average ± Deviation | $59.8 \pm 3.0$ | $\mathbf{66.0 \pm 3.0}$ |

## 5.2 Objective Evaluation with F1 Score

While ROC AUC was successfully used in [23] to evaluate their model, it is wise to consider a different way to evaluate our models due to the imbalance in the dataset. Using micro averaging that considers the dataset in total instead of treating all classes regardless the number of samples, we calculated the F1 scores of the previous model WC and our proposed model JF on each musical component and compiled the result in table 5–7.

With the highest increase observable on chord progressions, followed by tempi, and the highest final score at rhythm pattern, followed by chord progressions, we can see that despite the difference in order to that of ROC, with F1 score, JF also consistently scores higher compared to WC with lower deviation. This supports our claim that JF can replicate human preference better than WC.

Table 5–7: F1 scores in all 5 musical components

| Musical Components | WC F1(%) | JF F1 (%) |
|---|---|---|
| Chord Progressions | 23.4 | **32.1** |
| Rhythm Patterns | 35.8 | **38.2** |
| Instruments | 25.1 | **28.8** |
| Tempi | 23.3 | **28.9** |
| Drum Patterns | 21.8 | **24.5** |
| Average ± Deviation | $25.9 \pm 5.7$ | $\mathbf{30.5 \pm 5.1}$ |

## 5.3 Subjective Evaluation with Survey

Despite the overall increase in ROC AUC and f1 that was achieved, the final scores are still below 70% and 40% respectively. This means that the probability of the musical component recommendation by our models being different from what a real person would choose is still high.

To ensure that even with this mismatch, the resulting music composed with the recommendations from our models is still a good match to the lyrics input, we conducted a survey that compares three samples generated with Orpheus, one with setup done by a real Orpheus user, one with setup done by previous model MC, and one with setup done by proposed model JF.

### 5.3.1 Scoring the 3 Music Samples with Likert Scale

The 3 music samples are scored based on the music relevancy to lyrics on the Likert scale from 1 to 5, with 1 being a bad match and 5 being a good match. Separate surveys were conducted for each musical component with different music samples to allow respondents to focus on 1 specific change applied to the samples. This means that the 3 samples in each survey share the same lyrics and all setups, excluding the 1 musical component that is the subject of the survey.

We gathered 10 responses for each musical component and respondents were allowed to freely fill out at least 1 up to all 5 surveys. To make sure that there is no bias in scoring, we made sure that the file names of the music samples of the same model do not show any kind of similarities and put them in different orders depending on which musical component is used as a subject.

Table 5–8: Likert scores of compositions by Orpheus user, WC, and JF

| Musical Components | Original | WC | JF |
|---|---|---|---|
| Chord Progressions | 3.5(±1.08) | 2.7(±1.16) | 2.8(±1.03) |
| Rhythm Patterns | 2.8(±1.14) | 3.0(±1.41) | 3.0(±1.15) |
| Instruments | 2.3(±1.25) | 2.0(±0.94) | 3.0(±1.56) |
| Tempi | 3.6(±1.17) | 3.2(±1.03) | 3.4(±1.35) |
| Drum Patterns | 2.9(±0.88) | 2.7(±0.82) | 2.7(±1.06) |

From table 5–8, we can see that JF scores higher than WC on chord progression, instrument, and tempo. It scores the same in rhythm and drum pattern, with lower deviation on rhythm and higher on drum pattern. In other words, JF generally performs better in comparison with WC, except on drum pattern.

To gauge the quality of the surveys and the respondents themselves, we also asked the respondents if they can understand the difference between each sample, if they are confident with their answers, and their reasoning behind their answers. Looking at the graph in Fig. 5–1, unlike in other surveys, most people could not tell the difference between the 3 samples in drum pattern survey and almost half of them are not confident with their answers. In other words, JF performs better when the respondents can tell the difference and are confident.

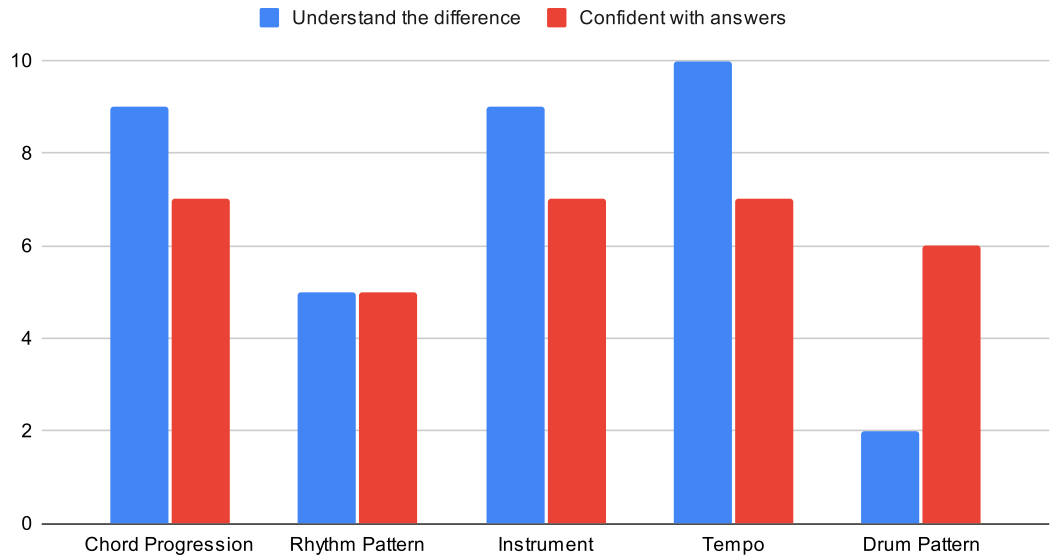


Fig. 5–1: Respondents understanding and confidence

Interestingly for rhythm pattern and instrument as shown in table 5–8, JF scores even higher than the original setup. As a conclusion, we can say that the objective results are generally reflected subjectively and found that JF can sometimes outperform the original setup.

### 5.3.2 Weighting the Respondent Scores Based on Their Reasoning

Looking at the survey responses, some respondents stated that they did not understand the difference between the three samples, were not confident in their answers, or both. Some of them had good reasoning behind their answers, while some others did not. Sometimes, we see reasoning that is irrelevant to the subject of survey, such as commenting on lyrics when only the drum patterns are changed, and so on. Considering this variation in respondents, we did a second evaluation with these answers as score weights ranging from 0 to 3 to see if there is any significant difference in the resulting weighted scores.

We give 1 point for each of these factors: $d_i = 1$ if the respondent understand the difference between the three samples, $c_i = 1$ if the respondent is confident with their answers, and $r_i = 1$ if the respondent is able to give relevant reason or $r_i = 0.5$ if their reasoning is not exactly relevant but not necessarily wrong, for example by commenting on how the music should have been in general instead of specifically commenting on the musical component used as subject in the survey. Considering that the number of respondent for each musical components survey $n = 10$, the sum of these factors is used as weight $w_i = d_i + c_i + r_i$ to the value of score $s_i$ and changes the score average $A_s$ calculation as follows:

$$A_s = \frac{1}{n} \sum_{i=1}^{n} w_i s_i \tag{5.1}$$

With these factors as weights, we can see some changes on table 5–9 in comparison with table 5–8. For chord progressions and tempi, we can see that there is a slight decrease in score of WC and an increase on JF. This pattern can also be seen on rhythm pattern where WC and JF originally score the same, making JF score higher this time. On instruments, there is no change on WC, but an increase on JF, giving it further lead among the three samples even though the score of the original samples also increased, albeit slightly.

Table 5–9: Weighted scores of compositions by Orpheus user, WC, and JF

| Musical Components | Original | WC | JF |
|---|---|---|---|
| Chord Progressions | 3.58(±1.04) | 2.67(±1.13) | 2.88(±0.99) |
| Rhythm Patterns | 2.80(±1.11) | 2.93(±1.38) | 3.02(±1.23) |
| Instruments | 2.39(±1.04) | 2.00(±0.83) | 3.35(±1.43) |
| Tempi | 3.60(±1.19) | 3.19(±1.00) | 3.53(±1.35) |
| Drum Patterns | 3.11(±0.70) | 2.71(±0.61) | 2.64(±1.01) |

However, we can see the opposite happening in drum pattern, resulting in WC scoring higher than JF. As previously mentioned in 5.3.1, specifically on drum pattern, most respondents stated that they did not understand the difference between the 3 samples and almost half were not sure of their answers. This might have been caused by the dominance of vocal and other instruments in the music samples, making it difficult for respondents to focus on the drum pattern itself, which explains the decrease on JF.

Overall, with the exception on drum patterns, JF has higher scores than WC either weighted or not. It sometimes even outperforms the original composition which setup was done by real Orpheus users. This means that not only can it replicate human selection better than previous model WC based on ROC AUC, JF can still offer decent, or sometimes better recommendations when the result deviates from what a person would choose.

## 5.4 How to Best Train and Evaluate Musical AI Models

In order to build a system that is capable of recommending appropriate musical components based on lyrics input, we took the model proposed in [25] and improved it by changing the pre-processing method and train the MLP model with a loss function suitable for imbalanced dataset. As we have argued in 2.3.2, while word2vec is viable to pre-process textual data such as lyrics, interaction between words are not considered in the calculation, which led us to use SBERT to create more appropriate semantics embedding from the lyrics.

Data imbalance was also not considered in [25], which led us to limit our scope by using only the top 10 dataset of chosen musical components, consequently excluding options with number of samples that is too low for training. In addition to that, we applied categorical loss function to improve training quality of the MLP models. We further expanded our research by experimenting it on not only one but five musical components as we saw the potential of this approach.

Although overall improvements can be seen by applying this approach during training and evaluation, both objectively and subjectively, we believe that it is possible to achieve better results. We would like to discuss three ways that can potentially better our approach as a reference for future research, which covers how to train and evaluate the models, and the model architecture itself.

### 5.4.1 Hard vs Soft Match

The training metrics used in training is the accuracy of the models in terms of predicting the same musical component option as chosen by the original composer, which means, when the model gives a prediction that is of a different option, it is given penalty for not giving the exact option it was expected to output. This may not be the best approach for this task for two main reasons.

For the first reason, let us take chord progressions as an example. As previously mentioned in 3.2.1, chord progression options in Orpheus are labeled with different names. It was also explained that each option consists of a sequence of chords. While it is true that each sequence is unique, they may still share the same chords in specific elements. By using the label instead of the element of sequence, we are risking the MLP model to get punished when it offers a similar sequence, just because it is not exactly what was expected.

This is true for other components as well. Just like chord progressions, rhythm and drum patterns can be further dissected into its elements. Instruments, as mentioned in [48] may sound similar with one another, making it possible to map them based on their similarity. Last but not least, tempo may be grouped better based on their typical genre representation as mentioned in [49] instead of strictly mapped in increments of 10 as we have done.

Next, let us review the survey result to discuss the second reason. We can see that altering the setup from the original composition may result in higher score in terms of how well it matches the lyrics. This is possible because Orpheus, our source of dataset (which also includes the test data), is a system that is available for public. It means that the quality of these composition depends heavily on how much musical knowledge the original composer has and that the generated compositions are by no means comparable to commercial music.

The fact that samples with with different setups may share similar score further supports our point that accuracy may not be the best training metrics for this task. It is a possibility that using training metrics that are less strict may result in estimation model that can perform better than what we have proposed.

### 5.4.2 Seq2Seq Approach for Chord Progressions

As previously discussed, each option of chord progression in Orpheus contain a unique sequence of chords. This sequential form makes it reasonable to apply sequence-to-sequence or seq2seq approach as proposed in [50] to solve this problem. The core idea is to treat the semantics embedding as a sequence of elements and use it to estimate how the chord will progress in a composition. This way, the resulting chord sequence will not be limited to what are present in the training data, making it possible to create unique progression that could result in a composition that is a better match to the lyrics input.

While this approach may not be suitable for other musical components, there might also be better approach that is more suitable for other musical components. By using different model training architectures that are specifically curated based on each musical components, we argue that it may be possible to achieve higher metrics scores during training and evaluation.

### 5.4.3 Interconnected Estimation Models with Multi-task LSTM

Based on the assumption that compositions that share a musical component may also share others as well, we argue that it might be possible to create an interconnected estimation model that can recommend all musical components at once instead of several of separate individual ones.

One way to achieve this is with multi-task LSTM (Long Short-Term Memory) that was introduced in [51]. It is a recurrent neural network (RNN) architecture widely used in deep learning. By also considering the relationship between the musical components in addition to that with the lyrics, the quality of model training can potentially be improved further.

# Chapter 6

# Conclusion

In this research, we noticed the necessity to automate musical components selection in lyrics-based automatic music composition system, such as Orpheus [7]. Referring to a number of related works, we considered the possibility of using machine learning to solve this estimation problem.

We took the approach by [25] and expanded it by introducing 4 musical components in addition to chord progression, which include: rhythm pattern, instrument, tempo, and drum pattern, after studying their relationship with music and lyrics. To better this approach, we used Sentence-BERT [31] instead of word2vec [28] to pre-process the lyrics into their numerical representation.

In addition to that, we noticed the imbalance in data extracted from Orpheus and its negative impacts on training. To exclude data with low number of samples, we compiled a top 10 dataset for each musical components, based on what options were chosen the most by Orpheus users. We utilized focal loss function introduced in [39] to minimize the overfitting caused by data imbalance.

By experimenting on pre-processing method, its pre-training corpus, and the loss function used in the training of the multi layer perceptron model used as a classifier in this research, we proved that our proposed model performs better than the previous model proposed in [25].

This claim is further supported by the evaluation results which was done objectively using receiver operating characteristic (ROC) and f1 score, and subjectively through surveys in which composition samples by Orpheus users, and the modified versions by proposed and previous models are compared.

Based on our findings, we have opened up the potential of future research which includes: soft-match based model training, seq2seq approach for chord progression estimation, and interconnected musical components estimation model.

# Acknowledgments

This endeavor would not have been possible without the supervision of professor Toru Nakashika and Yasuhiro Minami, for their guidance throughout my study. I am very grateful to be given the chance to work under their supervision despite my minimum knowledge in their field of expertise.

I would also like to express my deepest appreciation to professor Shigeki Sagayama and Gen Hori to let me take their work, Orpheus, as a research subject and their support in the progress of this research in the form of advice and insights, and to not give up on their support despite the countless miscommunication in my consultation due to language and cultural barrier.

Starting out as a total beginner in machine learning, I am deeply indebted to Mr. Takuya Takahashi for his continuous willingness and patience in teaching me everything from the very basic knowledge that has been very useful for the research. Words cannot express my gratitude to his presence.

Last but not least, I had the pleasure of working with with fellow members of Nakashika Laboratory, The University of Electro-Communications, Japan, and I would like to extend my sincere thanks to those who have participated in the surveys held for the purpose of evaluation.

This research was done under the support by Grant-in-Aid for Scientific Research (B) No. 21H03462 from Japan Society for the Promotion of Science (JSPS) and it was made possible with the scholarship provided by The Ministry of Education, Culture, Sports, Science and Technology (MEXT) Japan.

## Reference

[1] Ableton. https://www.ableton.com/. Last accessed at 28 July 2023.

[2] Image Line Software. FL Studio. https://www.image-line.com/. Last accessed at 28 July 2023.

[3] Apple. Logic Pro. https://www.apple.com/jp/logic-pro/. Last accessed at 28 July 2023.

[4] SOUNDRAW. https://soundraw.io/. Last accessed at 28 July 2023.

[5] Peter Crossley-Holland, Alexander L. Ringer, *et al.* “Music Theory & Compositions: musical composition”. Encyclopædia Britannica, last updated at 18 August 2022, Brittanica. https://www.britannica.com/art/musical-composition. Last accessed at 28 July 2023.

[6] OpenAI. “MuseNet”. Published at 25 April 2019, OpenAI. https://openai.com/research/musenet. Last accessed at 28 July 2023.

[7] Shigeki Sagayama *et al.* “Orpheus : An Automatic Music Composition System”, The journal of the Institute of Electronics, Information and Communication Engineers, Vol. 102, pp. 214–220 (2019). https://www.orpheus-music.org/. Last accessed at 6 August 2023.

[8] OpenAI. “Introducing ChatGPT”. Published at 30 November 2022, OpenAI. https://openai.com/blog/chatgpt. Last accessed at 28 July 2023.

[9] William L. Hosch. “Computers: machine learning”. Encyclopædia Britannica, last updated at 21 July 2023, Brittanica. https://www.britannica.com/technology/machine-learning. Last accessed at 28 July 2023.

[10] The Editors of Encyclopaedia Britannica. “Music Theory & Compositions: chord”. Encyclopædia Britannica, last updated at 23 July 2023, Brittanica. https://www.britannica.com/art/chord-music. Last accessed at 28 July 2023.

[11] Yong-Hun Cho, Hyunki Lim, Dae-Won Kim and In-Kwon Lee. “Music emotion recognition using chord progressions”, 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2588–2593 (2016).

[12] Peter Crossley-Holland. “Music Theory & Compositions: rhythm”. Encyclopædia Britannica, last updated at 24 July 2023, Brittanica. https://www.britannica.com/art/rhythm-music. Last accessed at 28 July 2023.

[13] Simon Dixon, Fabien Gouyon, and Gerhard Widmer. “Towards Characterisation of Music via Rhythmic Patterns”, International Conference on Music Information Retrieval (ISMIR), Proceedings (2004).

[14] Jack Allan Westrup, Theodore C. Grame, *et al.* “Musical Instruments: musical instrument”. Encyclopædia Britannica, last updated at 21 April 2023, Brittanica. https://www.britannica.com/art/musical-instrument. Last accessed at 28 July 2023.

[15] Aldona Rosner and Bozena Kostek. “Automatic music genre classification based on musical instrument track separation”, Journal of Intelligent Information Systems 50, pp. 363–384 (2004).

[16] Daniel Costa. “Music Theory & Compositions: tempo”. Encyclopædia Britannica, last updated at 5 May 2023, Brittanica. https://www.britannica.com/art/tempo-music. Last accessed at 28 July 2023.

[17] Martin McKinney and Dirk Moelants. “Ambiguity in Tempo Perception: What Draws Listeners to Different Metrical Levels?”, Music Perception 24, pp. 155–166 (2006).

[18] The Editors of Encyclopaedia Britannica. “Musical Instruments: drum”. Encyclopædia Britannica, last updated at 7 October 2022, Brittanica. https://www.britannica.com/art/drum-musical-instrument. Last accessed at 28 July 2023.

[19] Débora Corrêa, José Saito, and Luciano da F. Costa. “Musical Genres: Beating to the Rhythms of Different Drums”, New Journal of Physics 12 (2010).

[20] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. “ DeepBach: a Steerable Model for Bach Chorales Generation ”, arXiv (2017). https://www.flow-machines.com/history/projects/deepbach-polyphonic-music-generation-bach-chorales/. Last accessed at 6 August 2023.

[21] Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick, Julian McAuley. “Deep Performer: Score-to-Audio Music Performance Synthesis ”, arXiv (2022). https://salu133445.github.io/deepperformer/. Last accessed at 6 August 2023.

[22] Andrea Agostinelli *et al.* “ MusicLM: Generating Music From Text ”, arXiv (2023). https://google-research.github.io/seanet/musiclm/examples/. Last accessed at 6 August 2023.

[23] Hasan Oğul and Başar Kırmacı. “Lyrics Mining for Music Meta-Data Estimation ”, IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 528–539 (2016).

[24] Adriano A. de Lima, Rodrigo M. Nunes, Rafael P. Ribeiro, and Carlos N. Silla Jr. “ Nordic Music Genre Classification Using Song Lyrics ”, Natural Language Processing and Information Systems (NLDB), Lecture Notes in Computer Science 8455 (2014).

[25] Kazuki Shinohara. “ Automatic Chord Progression Setting Considering the Meaning of Japanese Lyrics in Automatic Composition ”, Meiji University Graduation Thesis (2018).

[26] Yihao Chen and Alexander Lerch. “ Melody-Conditioned Lyrics Generation with SeqGANs ”, arXiv (2020).

[27] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. “ SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient ”, arXiv (2017).

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “ Efficient Estimation of Word Representations in Vector Space ”, arXiv (2013).

[29] Joseph Akanya and Clement Omachonu. “ Meaning and Semantic Roles of Words in Context ”, International Journal of English Language and Linguistics Research 7, pp. 1–9 (2019).

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. “ BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding ”, arXiv (2019).

[31] Nils Reimers and Iryna Gurevych. “ Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks ”, arXiv (2019).

[32] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. “ Supervised Learning of Universal Sentence Representations from Natural Language Inference Data ”, arXiv (2018).

[33] Daniel Cer *et al.* “ Universal Sentence Encoder ”, arXiv (2018).

[34] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. “A Large Annotated Corpus for Learning Natural Language Inference”, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015).

[35] Adina Williams, Nikita Nangia, and Samuel Bowman. “ A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference”, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 1 (2015).

[36] Isao Sonobe. “sentence-bert-base-ja-mean-tokens-v2”, `https://huggingface.co/sonoisa/sentence-bert-base-ja-mean-tokens-v2`. Last accessed at 8 August 2023.

[37] Matthew Henderson *et al.* “ Efficient Natural Language Response Suggestion for Smart Reply ”, arXiv (2017).

[38] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. “ Multilayer perceptron and neural networks ”, WSEAS Transactions on Circuits and Systems 8 (2009).

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. “ Focal Loss for Dense Object Detection ”, arXiv (2018).

[40] Andrew P. Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms ”, Pattern Recognition 30, pp 1145–1159 (1997).

[41] Rensis Likert. “ A technique for the measurement of attitudes ”, Archives of Psychology (1932).

[42] Orpheus. “ Chord Progression Audio Samples”. Orpheus : An Automatic Music Composition System. https://www.orpheus-music.org/Orpheus-lib-harmony.php. Last accessed at 6 August 2023.

[43] Orpheus. “Rhythm Pattern Audio Samples”. Orpheus : An Automatic Music Composition System. https://www.orpheus-music.org/Orpheus-manual.php?q=rhythm. Last accessed at 7 August 2023.

[44] Orpheus. “ Instrument Audio Samples”. Orpheus : An Automatic Music Composition System. https://www.orpheus-music.org/Orpheus-lib-inst.php. Last accessed at 28 July 2023.

[45] Orpheus. “ Drum Pattern Audio Samples”. Orpheus : An Automatic Music Composition System. https://www.orpheus-music.org/open.php?id=201703. Last accessed at 28 July 2023.

[46] Takumitsu Kudo. “ MeCab : Yet Another Part-of-Speech and Morphological Analyzer ” (2005). https://api.semanticscholar.org/CorpusID:61584143. Last accessed at 28 July 2023.

[47] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. “ Fast WordPiece Tokenization ”, arXiv (2021).

[48] Pollastri Emanuele, Maurizio Longari, and Agostini Giulio. “ Musical Instrument Timbres Classification with Spectral Features ”, EURASIP Journal on Advances in Signal Processing, pp. 5–14 (2003).

[49] Mao-Yuan Kao, Chang-Biau Yang, and Shyue-Horng Shiau. “Tempo and beat tracking for audio signals with music genre classification”, International Journal of Intelligent Information and Database Systems (IJIIDS) 3, pp. 275–290 (2009).

[50] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”, arXiv (2014).

[51] Ralf C. Staudemeyer and Eric Rothstein Morris. “Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks”, arXiv (2019).

# List of Figures

# List of Tables

# Appendix A

# The Top 10 Chord Progression

For a quick reference, the sequence for each option that is included in the top 10 chord progression based on the list in [42] is as follows:

Table A–1: The sequences of top 10 chord progression

| Option Label | Sequence of Chord |
|---|---|
| pattern O | CM → Em7 → Am7 → Am7 → FM7 → Bm7 → Dm7 → G7 → CM → Em7 → Am7 → Am7 → FM7 → Dm7 → G7 → CM |
| pattern FF | CM → CM → CM → CM → CM → CM → CM → G7 → CM → CM → CM → CM → CM → CM → G7 → CM |
| pattern Q | Cm → G7 → G7 → Cm → Fm → Cm → G7 → Cm → Cm → G7 → G7 → Cm → Fm → Cm → G7 → Cm |
| pattern P | CM → GM → Am7 → Em7 → FM → Em7 → Dm7 → G7 → CM → GM → Am7 → Em7 → Dm7 → G7 → CM → CM |
| pattern H | Cm → GM → Cm → Cm → Cm → GM → Cm → Cm → EbM → BbM → Cm → GM → Cm → GM → Cm → Cm |
| pattern E | FM7 → E7 → Am7 → Am7 → FM7 → G7 → CM → Gm7 → FM7 → E7 → Am7 → Am7 → FM7 → G7 → CM → CM |
| pattern W | CM → G7 → CM → G7 → CM → FM → GM → GM → CM → C7 → FM → Fm → CM → G7 → CM → CM |
| pattern R | Am → E7 → Am7 → Am → FM7 → Dm7 → G7 → CM → Am → E7 → Am7 → Am → FM7 → Dm7 → G7 → CM |
| Pachelbel Kanon Ending | CM → GM → Am → Em → FM → CM → Fm → GM → CM → GM → Am → Em → FM → CM → Fm → GM |
| User Harmony zkrxx7 | Am → FM → GM → CM → Am → FM → GM → CM → Am → FM → GM → CM → Am → FM → GM → CM |

# Appendix B

# The Top 10 Rhythm Pattern

According to the samples available in [43], the rhythm patterns in Orpheus that are in the top 10 dataset are scored as follows:

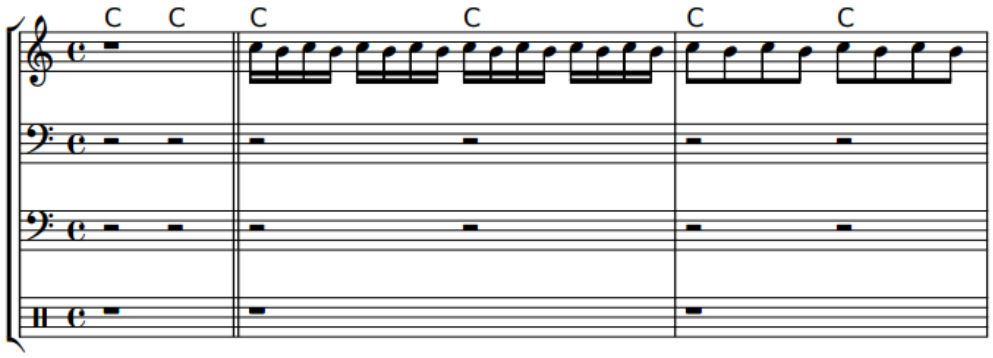


Fig. B–1: Musical score of pattern A32

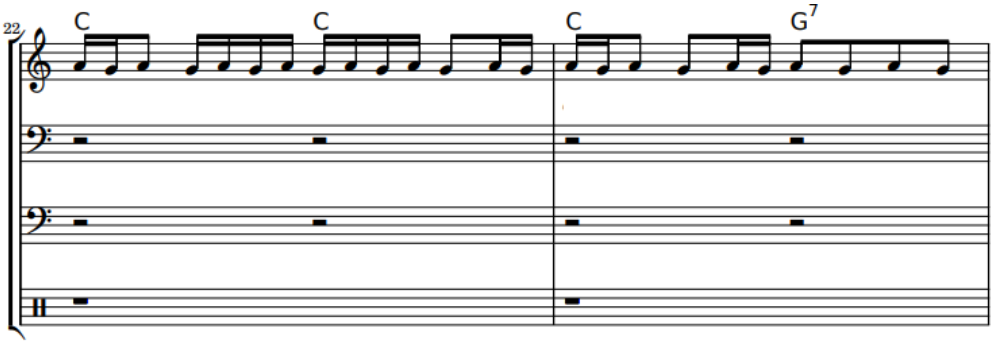


Fig. B–2: Musical score of syncopee 01ss

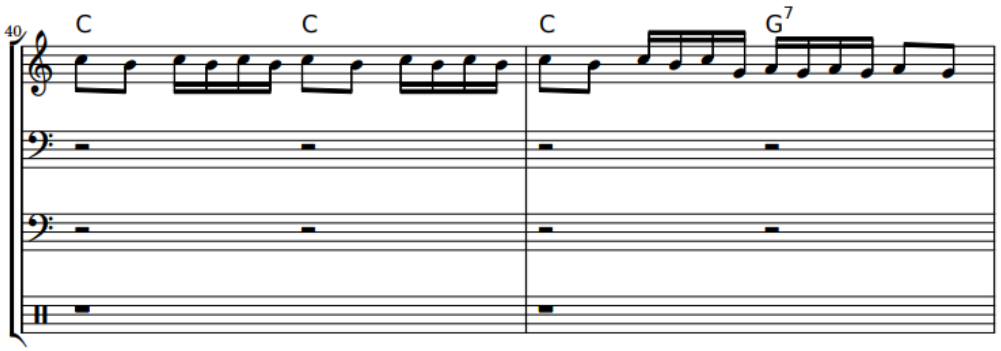


Fig. B–3: Musical score of love machine

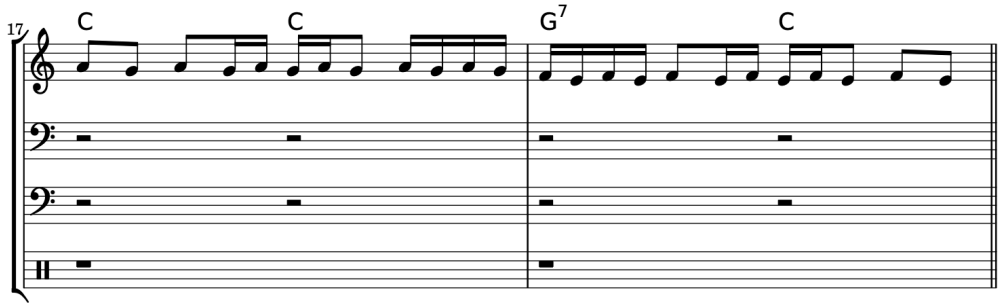


Fig. B–4: Musical score of school 1ss

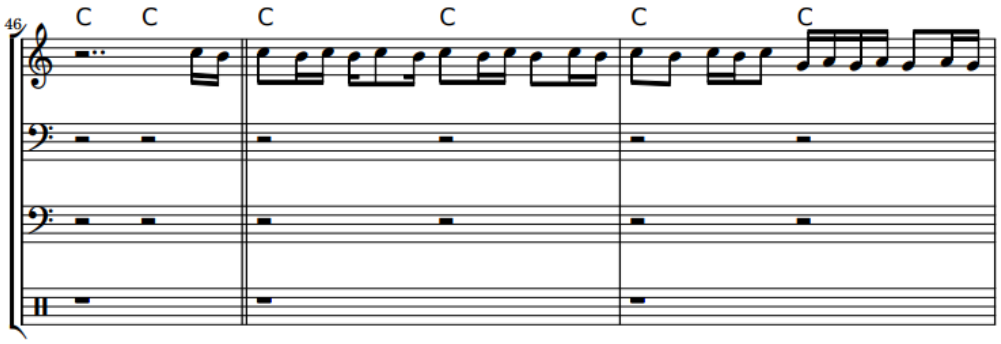


Fig. B–5: Musical score of Glass no shounen Kinkikids

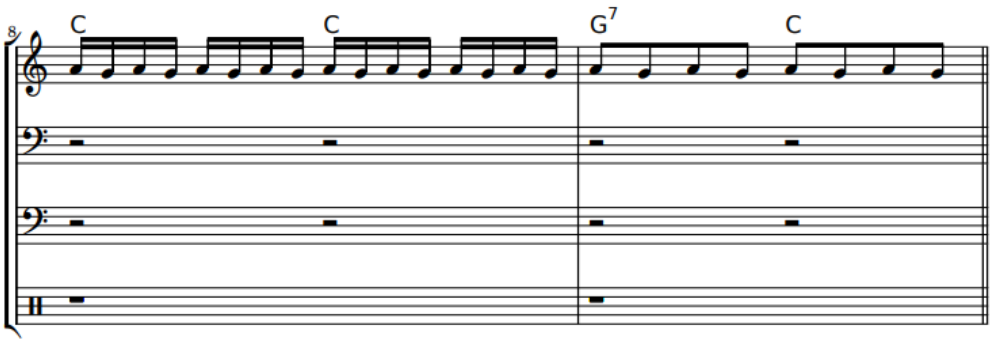


Fig. B–6: Musical score of pattern L32

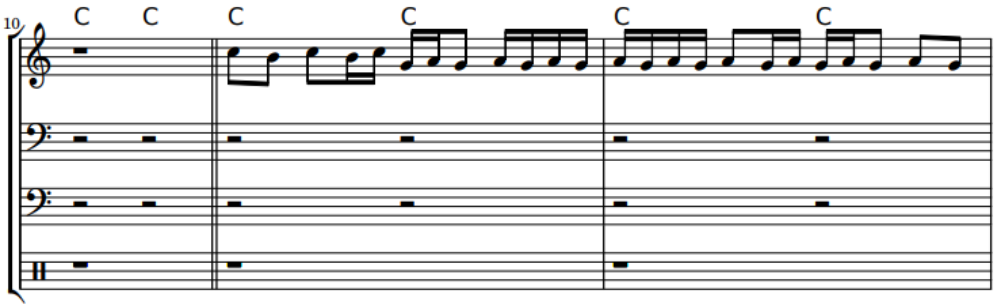


Fig. B–7: Musical score of pattern J32

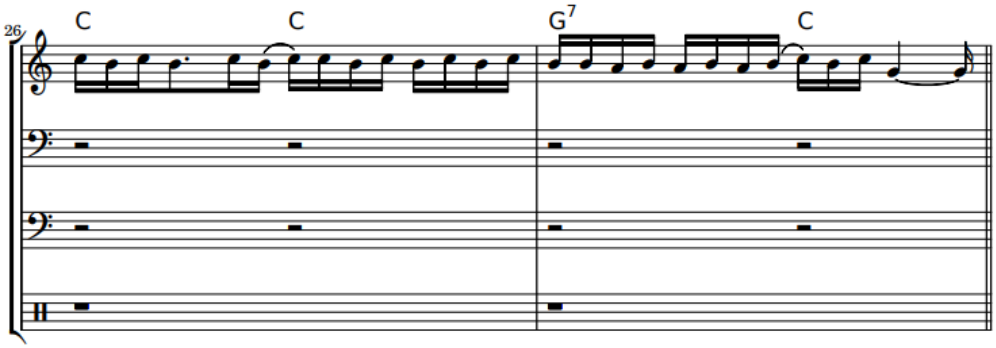


Fig. B–8: Musical score of sync-auf-3-16sf

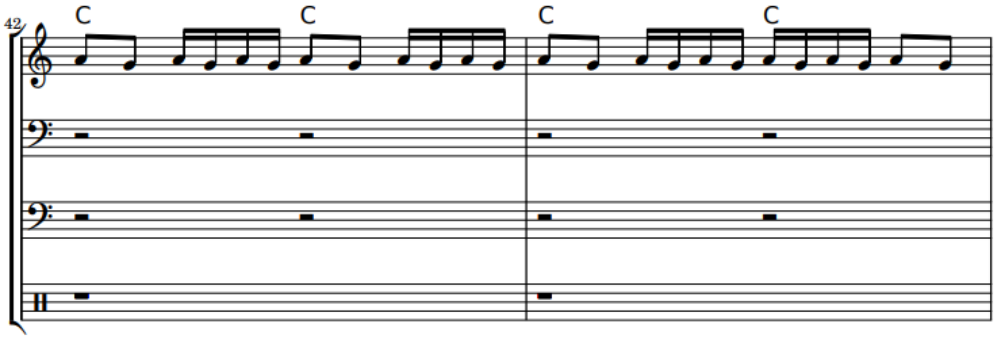


Fig. B–9: Musical score of chijo no hoshi 1ss

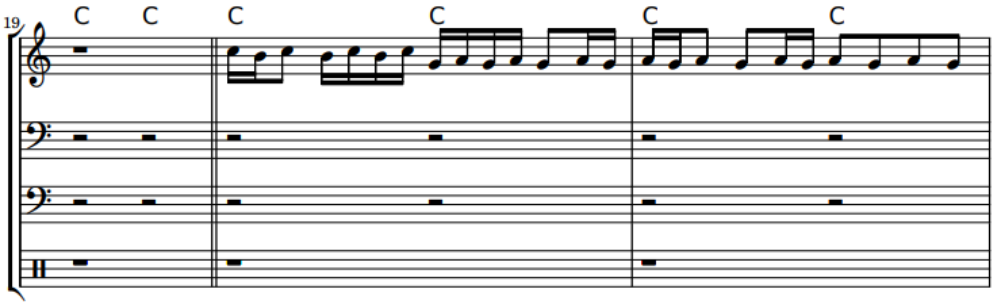


Fig. B–10: Musical score of march 110815ss

# List of Publications

- Mastuti Puspitasari, Mohamad Safrodin, and Fahim N.C. Bagar. “Real time pitch detection game to introduce local songs”, Journal of Fundamental and Applied Sciences 10(2S): Special Issue, pp. 311–322 (2018).
- Mastuti Puspitasari, Takuya Takahashi, Gen Hori, Shigeki Sagayama, and Toru Nakashika. “SBERT-based Musical Components Estimation from Lyrics Trained with Imbalanced “Orpheus” Data”, 音学シンポジウム (2023).
- Mastuti Puspitasari, Takuya Takahashi, Gen Hori, Shigeki Sagayama, and Toru Nakashika. “SBERT-based Chord Progression Estimation from Lyrics Trained with Imbalanced Data”, Computer Music Multidisciplinary Research (2023).