Computer Software and Media Applications 2025, 8(1), 11516.
https://doi.org/10.24294/csma11516

Article

LongMergent: Pioneering audio mixing strategies for exquisite music generation

Hong Lin, Xuan Liu, Chaomurilige, Kunyu Yang, Zheng He, Chang Xu, Zheng Liu*

College of Information Engineering, Minzu University of China, Beijing 100086, China
*** Corresponding author:** Zheng Liu, liuzheng@muc.edu.cn

CITATION

Lin H, Liu X, Chaomurilige, et al. LongMergent: Pioneering audio mixing strategies for exquisite music generation. Computer Software and Media Applications. 2025; 8(1): 11516. https://doi.org/10.24294/csma11516

ARTICLE INFO

Received: 16 February 2025
Accepted: 10 March 2025
Available online: 26 March 2025

Abstract: Artificial intelligence-empowered music processing is a domain that involves the use of artificial intelligence technologies to enhance music analysis, understanding, and generation. This field encompasses a variety of tasks from music generation to music comprehension. In practical applications, the complexity of interwoven tasks, differences in data representation, scattered distribution of tool resources, and the threshold of professional music knowledge often become barriers that hinder developers from smoothly carrying out generative tasks. Therefore, it is essential to establish a system that can automatically analyze their needs and invoke appropriate tools to simplify the music processing workflow. Inspired by the recent success of Large Language Models (LLMs) in task automation, we have developed a system named LongMergent, which integrates numerous music-related tools and autonomous workflows to address user requirements. By granting users the freedom to effortlessly combine tools, this system provides a seamless and rich musical experience.

Keywords: music generation; Large Language Models; audio mixing strategies

1. Introduction

The application of artificial intelligence in the field of music can help musicians and listeners experience and participate in music in entirely new ways. This application domain covers a variety of basic functions of music generation, generating music with multiple characteristics through the collaborative effort of various functions. Music understanding and generation are the results of the integration of multiple dimensions, which include many subprocesses (such as music retrieval, music style analysis, audio processing, etc.).

In the field of music understanding, Gómez and others proposed the standardization method of MPEG-7 audio and music description, which improved the standardization and interoperability of score transcription and music information retrieval [1]. However, the method relies on specific descriptors and features and is not suitable for all types of music or audio content; Meng et al. proposed a deep convolutional neural network (DCNNs) based music style classification model, which can automatically analyze music audio files, extract features, and classify music styles based on these features [2]. However, it requires a large amount of computing resources for training and inference, and its decision-making process lacks transparency.

In the field of music generation, Hadjeres and others proposed the DeepBach model for harmonic generation of Bach's choral music [3], but it is limited to producing melodies of a specific style; Chen and others proposed the HiFiSinger model, a high-quality singing voice synthesis system capable of generating high-

fidelity singing voices, but it requires a large amount of computing resources and training data [4].

Integrating the research findings of predecessors, it is not difficult to find that how to reduce the cost of computation is an urgent problem to be solved. At the same time, it is also necessary to take into account the unification of input-output formats and the integration of tasks [5]. Therefore, building a system that can efficiently integrate various methods of processing music according to the requests of users with different professional levels, and ultimately feedback to the users a music that can achieve the expected results in all aspects, is still a direction worth exploring.

The success of some other models (such as HuggingGPT) has provided us with a lot of valuable experience in exploring the development of a system that can assist in various music-related tasks [6]. In this paper, we will use the LLM model as the task scheduling hub to guide the cooperation between tasks.

Overall, it is not an easy task to meet the needs of a wide range of users to generate music, mainly reflected in the following three points:

a) The tools are scattered and dispersed. The diversity of music-related tasks leads to the dispersion of corresponding tools, which may be distributed across different platforms, such as open-source communities, software applications, or retrieval tasks hosted through Web APIs [7]. This means that to integrate these tools, it is necessary to address compatibility and standardization of interfaces between different platforms, otherwise, it is difficult to effectively integrate data from various platforms;

b) The complexity of music understanding. Music is a complex art form involving multiple levels such as melody, harmony, and rhythm [8]. LLMs need to be able to understand and process these complex musical elements, which requires the model to have a high level of musical theoretical knowledge and audio analysis capabilities;

c) Non-professional users find it difficult to generate high-quality long-text audio. Most of the current model's generative corpora are limited in scale, making it difficult to extract corpora that satisfy long-text audio in one go, thus causing most of the generated results to be limited by duration.

To address the aforementioned issues, we introduce LongMergent, a system specifically designed to meet these challenges. The main purpose of this system is to free users from being confined to a single music generation tool and to maximize their needs in terms of music performance. LongMergent can be understood as a central dispatch system that uses the mechanism of LLM as a controller and a multitude of sub-tools related to music generation to identify user inputs and provide processed results. The detailed scheduling process is shown in **Figure 1**. To ensure seamless operation between each sub-tool, LongMergent strictly stipulates the input and output formats of each interface, confirming that the output format of the previous interface can be recognized by the next interface. In addition, each step of LongMergent's output can be taken out as a music sample on its own, and all samples can be artificially modified to form perfect audio clips, which to some extent can also provide some convenience for users with a foundation in music.

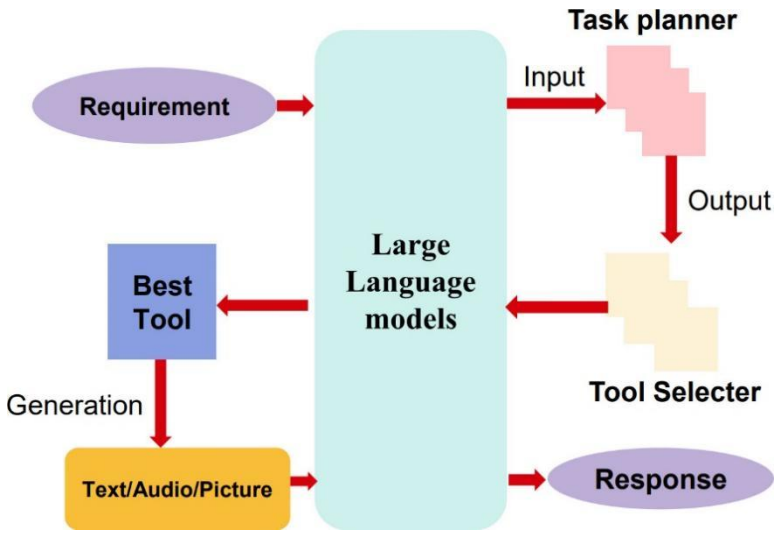

Figure 1. LongMergent system architecture and workflow.

In summary, LongMergent has made the following significant contributions:

a) It has created an integrated music processing system that provides a unified platform for handling music generation and understanding tasks. This integration reduces the need for users to switch between different tools and platforms, fostering collaboration among various music tools and enhancing work efficiency.

b) It leverages the powerful capabilities of LLMs to automate the processing of music tasks, including task planning, tool selection, and response generation. This enables the system to understand and break down complex user requests. Additionally, it has developed middleware capable of understanding and converting different music modalities (such as MIDI, audio, and sheet music), significantly reducing the difficulty and complexity of tasks.

c) It introduces an audio fusion mechanism that supports the merging of multiple short audio files into a long audio file while maintaining harmony. This allows non-professional users to fully express their needs through the system without the requirement of accessing a large-scale corpus.

2. Related work

2.1. AI-enabled music generation

Artificial Intelligence-Empowered Music Generation refers to the automated creation and generation of music pieces that align with specific styles, emotions, or requirements through artificial intelligence technologies. This provides creators with innovative tools and inspiration, enhances the efficiency and diversity of music production [9].

Generally, the field of music encompasses a variety of generation and understanding tasks, such as generating music from text descriptions [10], generating melodies from lyrics [3], classifying music [11], and processing audio. Among the many models, several have shown outstanding performance. For instance, Wu et al. proposed the Clamp model, which can retrieve symbolic music through text descriptions, effectively addressing cross-modal retrieval tasks and significantly

improving the accuracy of audio retrieval [12]. Wu et al. proposed a multi-channel non-negative matrix factorization (NMF) method for track separation [12,13]. Engel et al. proposed GANSynth, an adversarial network-based audio synthesis model capable of generating sounds of various musical instruments [14]. Hershey et al. proposed a method combining deep clustering and traditional networks for the separation and classification of music signals, effectively separating and classifying complex music signals by integrating different technologies [15]. Delving deeper, music is a hybrid art form that weaves together various elements such as chords, rhythm, and melody to synthesize vivid and harmonious content [16]. Previous works scored well in generating singular, straightforward styles of content, but often struggled with complex music tasks, neglecting one aspect for another.

From a deeper analytical perspective, music generation necessitates handling multi-dimensional artistic elements such as chord progressions, rhythmic patterns, and emotional mappings. Early models like DeepBach were constrained by single-style generation capabilities, while HiFiSinger relied heavily on high computational resources. Although Music Transformer captures long-range dependencies through self-attention mechanisms, its monolithic architecture struggles to dynamically adapt to complex task requirements. In contrast, LongMergent employs an LLM-driven tool orchestration framework that integrates diverse music tools via standardized interfaces (see **Table 1**). This architecture supports cross-modal transformations, significantly enhancing system flexibility. For instance, when processing a request to "generate a rock-style song based on World Cup themes", Music Transformer would require training a specialized model again, whereas LongMergent can dynamically invoke tools like ChatGPT for lyric generation, ROC for melody creation, and Basic-merge for mixing. This reduces the total workflow time from traditional hours to tens of minutes.

Therefore, in this paper, we will discuss how to unify music data formats and utilize Large Language Models to autonomously complete music generation tasks.

2.2. Natural language large models

The field of natural language processing is undergoing a revolutionary shift due to the emergence of Large Language Models (LLMs). Recently, LLMs have shown great performance in solving natural language processing (NLP) tasks, gaining favor among researchers. The immense potential of LLMs has also inspired and directly promoted many emerging technologies, such as context learning [17], natural language instruction learning [18], and multimodal learning [19]. With the widespread application of these technologies, the processing capabilities of LLMs have been further enhanced. Based on these LLM capabilities, many researchers have expanded the scope of LLMs to various topics, such as text generation and image processing. They draw on the idea of positioning LLMs as central schedulers, orchestrating various domain-specific expert models to solve complex artificial intelligence tasks. As a result, many models centered on LLMs have emerged. For instance, Liu et al. explored the application prospects of LLMs in music processing and proposed a multimodal music understanding and generation framework based on LLMs [20]; Chen et al. studied the application of Audio LLMs in voice quality assessment,

enabling Audio LLMs to perceive voice quality in a human—like manner for the first time [21]; Zeng et al. (2024) designed a trainable pipeline for LLM function calling in enterprise scenarios, enhancing the response and operation efficiency of large models in specific business settings [22]; Julian et al. combined LLMs with function calling technology to build smarter and more human—centred AI Agents.

However, existing models focused on music generation often produce rigid audio with monotonous melodies. LongMergent aims to break new ground in three ways:

a) Music Modality Adaptation: It will create middleware to support MIDI, audio, and sheet music conversions, resolving the issue of fragmented music data formats.
b) Dynamic Weight Allocation Mechanism: An context—aware weight adjustment algorithm will be introduced during audio mixing to ensure the generated long—audio is coherent.
c) Domain—specific Knowledge Infusion: By leveraging the MELOLIB database, LongMergent will infuse music theory into LLMs, significantly boosting their accuracy in chord - matching tasks.

3. LongMergent

LongMergent is a comprehensive system that, based on ensuring the basic capabilities of Large Language Models (LLMs), incorporates datasets and toolkits from multiple sources to collaborate and complete tasks. LongMergent designs an autonomous workflow empowered by LLMs, including three key skills: Task planner, tool selector, and response generator. These skills, combined with the music-related tools that constitute the task executor, form a multifunctional system capable of executing various applications. In this section, we will carefully analyze each level of this system, including task allocation planning, autonomous workflow, music generation process, and music synthesis process, to further explore its functions and contributions to the field of music processing.

3.1. Task allocation and tool coordination

Table 1 provides a comprehensive overview of the music-related tasks and representative tools currently collected in LongMergent. The overall tasks can be summarized into the following three parts:

a) Auxiliary Search Tasks. These tasks include internet searches and matching various audio processing toolkits. Internet searches involve using the Google API for text searches to select song lyrics that meet user needs for further creation, and using the Spotify API for music searches to locate specified musical works and extract their musical features.
b) Music Understanding Tasks. Through the analysis and processing of audio data, the LongMergent system can categorize music into different styles, genres, emotions, etc., such as distinguishing whether a song is pop, rock, or classical. Other important functions at the understanding level include track separation, sentiment analysis, and style identification.
c) Music Generation Tasks. With tools like ROC, the LongMergent system can transform input text descriptions into corresponding musical notation, thereby

generating complete musical works, such as creating a piece of soothing and beautiful music that matches a description of a natural landscape. Other indispensable functions include vocal synthesis, timbre conversion, accompaniment generation, and song synthesis. The quality of the music generated by the LongMergent system is the most direct feedback from this task.

Table 1. Music processing tasks.

Task	Input	Output	Task Type	Example Tool
lyric-generation	text	text	Auxiliary	ChatGPT
lyric-to-melody	text	Symbolic music	Generation	ROC
lyrics-to-audio	text	audio	Generation	DiffSinger
timbre-transfer	audio	audio	Generation	DDSP
accompaniment	audio	audio	Generation	GetMusic
audio-mixing	audio	audio	Generation	Basic-merge
music-classification	audio	text	Understanding	Wav2vec2
music-separation	audio	audio	Understanding	Demucs

Additionally, LongMergent illustrates the use of three primary data formats within the system: i) Text, which includes lyrics, genres, and any other attributes related to music; ii) Sheet Music, represented in the form of MIDI files; iii) Audio.

3.2. Autonomous workflow

3.2.1. Skills

The skill module is composed of a task planner, tool selector, and response generator.

The task planner is the brain of the system, responsible for breaking down users' natural language requests into executable sub-tasks. It leverages the context learning capabilities of Large Language Models, combined with preset task planning templates and a wealth of example data, to accurately understand user needs and generate reasonable task decomposition schemes. For example, for a request like "compose a song with a rock style", the task planner might break down the task into sub-tasks such as lyric creation, melody generation, and arrangement (selecting rock-style instruments and rhythm patterns), and determine their execution order and data interaction methods.

The tool selector is responsible for choosing the most suitable tools for each sub-task from a multitude of available music tools. It maintains a tool library that includes information on music tools from various sources (such as Hugging Face, GitHub, various APIs, etc.). When selecting tools, it matches the current sub-task requirements with the tool attributes in the tool library, obtains the most suitable tool recommendations through targeted prompting and inquiries with the large language model, and records the reasons for selection.

The response generator is responsible for integrating the intermediate results after each sub-task execution into a coherent and understandable response returned to the user. It utilizes the text generation capabilities of Large Language Models to organize and package data based on the task processing determined by the task planner and the

tool execution results provided by the tool selector. It provides considerate feedback by reasonably organizing data in different formats and types (such as text, audio, sheet music, etc.), meeting users' expectations for the results.

3.2.2. Plugins

Plugins play a role in handling special functions throughout the system, such as extracting music spectral features and converting music styles, which require a variety of plugins to accomplish. The main plugins are as follows:

Hugging Face Music Wav2vec2. The Wav2vec2 model provided by the Hugging Face platform plays an important role in the field of music processing, especially in audio feature extraction and processing. It can extract valuable feature information from music audio, such as vocal characteristics and musical rhythm features, providing foundational data for tasks like music classification and transcription.

DDSP Module. The DDSP module provides LongMergent with powerful audio processing capabilities, especially in timbre conversion and audio synthesis. It achieves flexible adjustment and synthesis of musical timbres through differentiable processing of audio signals. In timbre conversion tasks, DDSP can analyze and model the timbre of input music and then convert it to the target timbre according to user needs, such as converting piano timbre to violin timbre or creating unique hybrid timbres.

API Google Audio Editing Module. Through audio-related APIs provided by Google, the LongMergent system can achieve efficient audio mixing and editing operations. These operations are crucial for music production and editing, helping users to flexibly handle musical materials. In music mixing tasks, users can use this API to mix multiple audio tracks (such as performances of different instruments, vocals, etc.), adjust the volume, balance, spatial position, and other parameters of each track to create rich and harmonious musical effects. At the same time, in audio editing, it can precisely cut, splice, and edit music to meet users' personalized needs for music segments.

User-Defined Solutions. To meet users' personalized needs, the LongMergent system allows users to customize solutions. Users can develop or integrate their own music processing tools or algorithms according to their specific needs and creativity and integrate them into the system.

Other Built-in Features. The system itself has some built-in music processing functions, which are based on the core algorithms and technologies of LongMergent and can meet some common music processing needs.

3.2.3. Applications

The specific application modules are the actual output modules of the entire system; all orchestration is aimed at invoking these powerful tools to produce outstanding works. The main application modules are as follows:

a) Song Creation Module. When a user requests to create a song, the task planner breaks down the task into sub-tasks such as lyric creation, melody generation, arrangement, and mixing. The tool selector chooses appropriate tools based on the sub-task requirements, such as selecting a lyric generation model from Hugging Face, a melody generation algorithm from the backend generation tasks, and suitable arrangement and mixing tools from the plugins. The task executor is

responsible for executing these tools, generating content for each part, and finally, the response generator integrates these contents into a complete song and presents it to the user. With this feature, users can obtain a complete song that meets their requirements through simple natural language commands, such as "Help me create a romantic love song", without the need for professional music creation knowledge and skills, greatly lowering the barrier to music creation.

b) Music Analysis Feature. For music analysis requests, the task planner determines the specific direction of analysis, such as music style analysis, sentiment analysis, and instrument analysis, and breaks it down into corresponding sub-tasks. The tool selector chooses appropriate analysis tools, such as using Wav2vec2 for audio feature extraction, followed by music classification algorithms in the backend's understanding tasks for categorization. The response generator presents the analysis results in an intuitive way to the user. Users can easily understand various attributes and characteristics of music, such as the style and emotional tendencies of unfamiliar music, which is significant for music appreciation, research, and education.

c) Timbre Conversion Feature. In timbre conversion tasks, the task planner identifies the source and target timbres for conversion, the tool selector chooses timbre conversion tools like DDSP, the task executor performs the timbre conversion operations, and finally, the response generator displays the converted music. Users can easily transform the timbre of music with this feature, creating unique musical effects to meet the diverse needs in music creation and production.

d) Score Transcription Feature. For score transcription tasks, the task planner takes the music audio file as input, the tool selector chooses suitable score transcription tools, the score transcription feature in the backend's understanding tasks is executed to convert audio into sheet music, and finally, the response generator displays the transcribed score.

e) Mixing Feature. In mixing tasks, the task planner determines the audio tracks and parameter requirements for mixing, the tool selector chooses mixing tools like the API Google Audio Mixing Editing, the task executor performs the audio mixing operations, and finally, the response generator displays the mixed music effect. Users can professionally mix multiple audio tracks to create the desired musical effects, meeting the needs for music mixing in music production and live performance scenarios, and enhancing the overall quality of music.

3.3. Music generation

The LongMergent system has various capabilities for generating music. Below, taking a specified set of lyrics as an example, we analyze how the system generates music step by step.

3.3.1. Lyrics to melody

Suppose the user inputs the lyrics "Sky blue awaits the misty rain, and I'm waiting for you", the LongMergent system will determine that the user is using default parameters. The script first parses command line arguments to obtain the paths for the lyrics file, chord progression file, database, as well as parameters for whether to output

debug information and whether to automatically set the key based on the emotional tone of the lyrics.

Next, it further analyzes the language of the lyrics. Based on the first two characters of the chord file, if it's in Chinese, it generates the melody according to the common style of Chinese songs; if it's in English, it follows the common style of English songs. In this example, Chinese is detected, so the melody is generated in the style of Chinese songs.

For each line of the lyrics, the LongMergent system processes it differently based on structural values. It first selects melody fragments from the database based on conditions such as major key, whether it's a chorus, lyric length, the last note of the previous musical segment, and chords. Selection follows the scoring function S_i, defined as:

$$Si = \alpha \times sim(C, Ci) + \beta \times (\frac{L}{len(f_i)}) + \gamma \times emo(L) \quad (1)$$

Here, S_i is the i-th melody fragment's score; α, β, and γ are weighting parameters. sim(C,C_i) measures the similarity between the current chord C and the fragment's chord C_i. L is the current lyric length, len(f_i) is the fragment's length, and emo(L) indicates the matching degree of emotion between the lyrics and fragment. The system selects the most suitable melody fragment based on the S_i value.

If the lyrics are short and there are candidate fragments, it calculates the language model score for further filtering; if no suitable fragments are found or the lyrics are too long, it splits the lyrics and repeats the search and splicing process, but the system will forcibly terminate the process after 10 repetitions to prevent falling into an infinite loop.

If a match is successful, the system optimizes the selected melody fragments, including avoiding fragment overlap, adjusting the starting position of notes, removing rests and cadences, etc., to obtain the optimized melody. It also adjusts the alignment of chords and melody. The optimization function O is expressed as:

$$O(\mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_\mathrm{n}) = \sum_{\mathrm{i}=1}^{\mathrm{n}-1} (Harmony(m_i, m_{i+1}) + Smoothness(m_i, m_{i+1}) \quad (2)$$

Here, m_1,m_2,…,m_n are a series of melody fragments. Harmony(m_i,$\mathrm{m}_{\mathrm{i}+1}$) denotes the harmony between two consecutive fragments, and Smoothness(m_i,$\mathrm{m}_{\mathrm{i}+1}$) indicates their smoothness. This function minimizes dissonance and maximizes smoothness between melody fragments.

After confirming that all optimization processes have been carried out, the system will save the generated melody (including all processed melody fragment splicing results), chords, song name, and lyrics (along with a list of lyric sentence lengths) as a MIDI file, completing the conversion process from lyrics to melody. The generated MIDI file is the melody generated based on the input lyrics. Depending on user requirements, it also decides whether to produce the corresponding WAV file. The detailed generation process is shown in **Figure 2**.

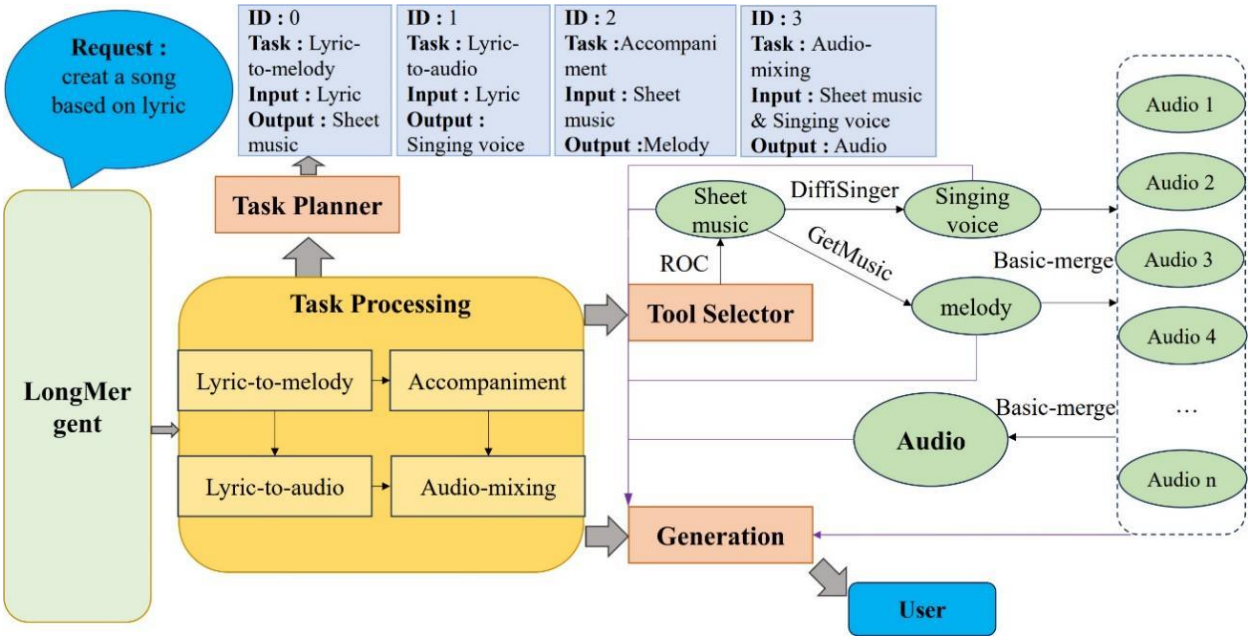

Figure 2. Song creation workflow in the LongMergent system. Illustrating the process where the agent's task planner coordinates the tasks of lyric-to-melody (using ROC), lyric-to-audio (using DiffiSinger), and audio-mixing (using Basic-merge) to generate a song based on the input lyric, showing the task IDs, inputs, tools, and results at each stage.

3.3.2. Lyrics to audio

The LongMergent system does not directly generate a complete piece of music; instead, it generates melody and vocals separately and then merges them onto the same audio track. Thus, the previous step of generating a melody is just one part of the process. Once the melody is confirmed to be error-free, the next step—generating the vocal accompaniment—is executed.

First, it is essential to confirm the availability of the necessary components, including phoneme encoders, speaker mapping dictionaries, and model-based vocoders, among other foundational elements. Next, the input is processed, with different preprocessing operations based on the type of input (defaulting to word-level). In this example, using word-level input, the LongMergent system invokes the preprocess word-level input method. This method first replaces any polyphonic characters in the lyrics and then converts the lyrics into a sequence of phonetic sounds (pinyin). It then processes the information regarding the musical notes and their durations within the lyrics. For each word, its pinyin is translated into a sequence of phonemes, aligns the musical notes with the phoneme sequence, handles connected speech, and ensures that each phoneme has corresponding note and duration information. Finally, the phoneme sequence is encoded into the model's input format, the musical notes are converted into MIDI IDs, the note durations are converted into floating-point numbers, and a dictionary item containing all relevant information (such as project name, text, phonemes, speaker ID, pitch MIDI values, note durations, and connected speech markers) is constructed.

After confirming that the necessary synthetic elements are available, the next step is to combine these elements into a complete vocal sequence. The LongMergent system uses a model to infer an intermediate representation, which is then converted

into an audio waveform using the HiFiGAN vocoder. Finally, the generated audio file can be saved.

3.3.3. Audio mixing

After successfully generating the melody and vocals, the subsequent task is to finalize the process by organically merging the melody and vocals to obtain a harmonious piece of music. The LongMergent system calls upon the basic-merge module to perform the audio mixing operation.

The LongMergent system passes the previously generated melody and vocals to the basic-merge model in the form of digital signals. The mixing rule involves adding corresponding sample points of two audio signals, $A = [a_1,a_2,\ldots,a_n]$ and $B = [b_1,b_2,\ldots,b_n]$, with certain weight coefficients. If the lengths of the signals are the same, no additional processing is required; otherwise, alignment processing is performed, padding the end of the shorter audio signal with zeros until both signals are of equal length. The sample points of the mixed audio signal are represented as:

$$c_i = \omega_1 a_i + \omega_2 b_i \tag{3}$$

where ω_1 and ω_2 are weight coefficients that users can adjust according to their needs to emphasize specific aspects.

Beyond simple additive mixing, the model also supports the integration of other audio characteristics, such as volume and channel information. For instance, normalizing the volume of the two audio signals before mixing ensures that the resulting audio does not have excessively high or low volumes. If the input audio is stereo (with left and right channels), the model performs more complex mixing operations based on channel information, such as mixing the left channel of one audio with the right channel of another. However, regardless of the focus, the primary principle of fusion is to obtain a harmonious piece of music in any circumstance without compromising audio quality for the sake of highlighting a particular feature.

After processing by the basic-merge model, the final audio data that meets the user's requirements can be obtained, and this data can then be saved as a new WAV file. This step represents the final phase in transforming lyrics into a complete musical composition.

4. Experiment and evaluation

The following is a brief introduction to the basic setup of this experiment.

4.1. Database and musical representation

We utilize the widely adopted MELOLIB database for our experiments. The MELOLIB database is a music resource library specifically constructed to support tasks based on generating melodies from lyrics. It sources from a vast collection of musical works that have been gathered, organized, and preprocessed, encompassing a multitude of music styles, genres, and emotional expressions. This aims to provide a rich variety of materials for melody generation, ensuring that the generated melodies can accommodate the demands of different lyrics. Additionally, it offers numerous forms of musical representation, such as the arrangement of note pitches and timelines, whether a melody fragment is a chorus, and the chord representation of melodies,

precisely meeting users' needs for specific characteristics. The details of this database are shown in **Table 2**.

Table 2. Detailed information of the MELOLIB database.

Statistical Item	Value
Average Track Length	3 min 15 s (standard deviation ± 45 s)
Genre Distribution	Pop 65%, Rock 20%, Electronic 10%, Other 5%
Annotation Dimensions	Melody/Chords/Emotion/Structure/Instruments
Data Source	Public Music Platforms (Spotify) and Professional Production Team
Average Chord Complexity	4.2

In this experiment, we will explore the quality of long-text audio synthesized using the LongMergent model and its similarity to individual sub-audio files. These sub-audio files are generated by the LongMergent system according to various style requirements, and the final long-text audio is assembled from different short-text audio files based on permutation combinations. We will set up four types of music styles, with five short-text audio files (not exceeding 30 s) for each style, and the long-text audio is synthesized from 2–3 short-text audio files of the same style. Consequently, we will assess the quality of the long-text audio based on corresponding evaluation metrics.

4.2. Implementation details

We input the desired music style into the dialogue interface, where the task planner invokes an LLM to parse our requirements and feeds back to the tool selector in the form of keywords. The tool selector determines the tasks as lyric-generation, lyric-to-melody, lyric-to-audio, and audio-mixing based on the received keywords, thereby dispatching tools such as ChatGPT, ROC, DiffiSinger, and basic-merge to accomplish the tasks. Finally, the response generator provides feedback on the results of each process in the form of text or WAV files. This describes the generation process of short-text audio files. Should we continue to synthesize long-text audio, we must re-express our needs in textual form in the dialogue box and upload the sub-audio files that require integration. Ultimately, we can obtain long-text audio files in six different styles.

4.3. Compared models

We first compare the long-text audio generated by the LongMergent system with the corresponding short-text audio to verify the integrity of its fusion. Subsequently, we will horizontally compare this data with corresponding data from other models that have outstanding performance in this field to comprehensively assess the quality of the generated music. The models with notable performance we have collected are as follows:

a) Music Transformer. By introducing self-attention mechanisms, it can generate high-quality, structured musical works, particularly excelling in capturing long-term dependencies and complex melodic patterns [23].
b) Transformer-XL. By incorporating long-range dependency modeling and more

efficient memory mechanisms, it effectively enhances the coherence and complexity of generated music, especially in long-section music creation [24].

c) Longformer. By employing local attention mechanisms, it effectively extends the model's ability to handle long sequences, making the generation of music over extended time spans more efficient and coherent [25].

d) Museformer. By combining fine-grained and coarse-grained attention mechanisms, it excels in handling long music sequences, capable of generating high-quality and well-structured music [5].

All compared models are set with fixed hyperparameters as Museaker. Due to memory constraints, we cannot batch generate long-text audio at once; instead, we divide each song into multiple samples during training and apply the model to generate long sequences during validation and inference to test its generalization on long music sequences.

4.4. Objective evaluation metrics

We will objectively assess our experimental results based on the following two metrics:

a) Perplexity (PPL). Perplexity is a measure of how well a language model predicts samples and is commonly used in natural language processing. The formula for PPL is as follows:

$$\mathrm{PPL} = \exp\left(\frac{1}{N}\sum_{\mathrm{i}=1}^{\mathrm{N}} \log P(x_i|x_1, x_2, \ldots, x_{i-1})\right) \tag{4}$$

where N is the length of the audio sequence, and $\mathrm{P}(\mathrm{x_i} \mid \mathrm{x_1},\mathrm{x_2},\ldots,\mathrm{x_{i-1}})$ is the probability of the model predicting the i-th note. The more accurate a model's prediction, the lower its perplexity. In the field of music, this metric is used to gauge a music model's ability to accurately reproduce data based on test data. In layman's terms, when we provide a high-quality audio segment that aligns with human auditory aesthetics as a test set, the higher the probability that the model generates this audio segment, the lower the model's perplexity is considered, and the better the model performs. For the LongMergent model, PPL reflects the probability of the t-th note occurring given the previous (t - 1) notes. This metric evaluates the model's generation performance across different time spans, helping assess its generalization and stability when handling long music sequences. We will measure the perplexity for audio of different lengths.

b) Similarity Error (SE). Similarity Error is a measure of the error between the output of a generative model and the target data, commonly used to evaluate the quality of models in generative tasks. The formula for SE is as follows:

$$\mathrm{SE} = \frac{1}{N}\sum_{\mathrm{i}=1}^{\mathrm{N}} |y_i - \hat{y}_i| \tag{5}$$

where y_i denotes the target data, and $\hat{y}_i$ represents the data generated by the model. In music, similarity error measures the degree of match between notes, lyrics, and other content. A lower SE indicates that the generated music is closer to the actual musical structure and patterns. For the LongMergent model, the SE value assesses the similarity between generated and target music. By comparing samples of varying

styles and themes, it evaluates the model's generation performance across different music styles, helping understand its ability to capture style features and produce high-quality music.

4.5. Subjective evaluation metrics

The most efficient and primitive method for assessing the quality of generated music is human listening tests. We invited five listeners with professional music knowledge and three without professional knowledge to rate our generated results. Participants will score these music segments on the following subjective metrics from 1 (lowest) to 10 (highest):

a) Musicality. Whether the generated music is emotionally rich and soulful like music composed by human musicians, rather than a mechanical combination of individual notes.
b) Short-term structure. Whether the generated music shows good structure in adjacent segments, mainly reflected in repetition and continuity.
c) Long-term structure. Whether the generated music shows good structure in segments that are far apart, such as repetitive segments or segments that echo previous parts.
d) Overall. An overall score based on the above criteria.

5. Results and discussions

Table 3 illustrates the detection results of similarity error across five themes in three distinct musical styles (pop, jazz, and classical). The generation results for each style were created with reference to the corresponding themes in the sample group, with lower values indicating higher similarity and lower error rates.

Table 3. Objectively evaluate the perceived similarity error result.

Theme	Popular Style	Jazz Style	Classical Style
Mountain	3.36	3.46	3.66
River	4.41	4.29	3.84
Birds	3.24	2.39	3.32
Pet	1.45	2.16	1.57
Auspicious	1.81	1.54	3.42
Average	2.85	2.77	3.16

Based on the results presented in the table, we can draw the following conclusions:

a) The model exhibits the smallest overall mean *SE* values in the jazz style group, suggesting that it performs relatively well in capturing the characteristics of jazz music;
b) Although the model's overall mean *SE* values in the pop style group are not as low as those in the jazz group, it outperforms the jazz group in certain themes, indicating that the model is more adept at interpreting specific themes (such as pets) using pop music styles;
c) The model's performance in the classical style group significantly lags behind

the other two styles, implying that the model faces challenges in understanding and reproducing the complexity of classical music.

Building on these findings, we compared the *SE* values of the LongMergent model with those of other models, as detailed in **Table 4**. The *SE* values for the LongMergent model are the average results of the three groups' means.

Table 4. Comparison of SE values among various models.

	Music Transformer	**Transformer-XL**	**Longformer**	**Museformer**	**LongMergent**
SE(%)	2.49	15.66	5.25	0.95	2.92

a) It is evident from the comparison results that our LongMergent model has a distinct advantage over some other high-performing models in terms of similarity error, with only a small gap when compared to the best-performing models. The Music Transformer's self-attention mechanism allows the model to consider information from the entire sequence when generating music, thus providing more accurate outputs than other models overall. However, while its generalization capability is strong, it struggles to outperform on specific themes, where LongMergent shows better performance. The Longformer focuses more on specific parts of the sequence, which may lead to oversights in overall control. The comprehensive comparison results indicate that the LongMergent system's performance in generating music that matches the given input is on par with other high-quality models.

b) Furthermore, we conducted perplexity tests on the model's generated data. We compared multiple groups based on sequences of three different lengths (1024, 5120, and 10240), all of which were assembled from varying short-text audio files. Lower PPL values indicate lower perplexity and better performance. The specific comparison results are presented in **Table 5**.

Table 5. Results of perplexity in objective evaluation. The numbers in parentheses represent the sequence lengths of the audio.

	Music Transformer	**Transformer-XL**	**Longformer**	**Museformer**	**LongMergent**
PPL(1024)	1.66	1.64	1.65	1.64	1.51
PPL(5120)	1.77	1.45	1.46	1.41	1.41
PPL(10240)	2.55	1.43	1.45	1.35	1.34

Upon comprehensive analysis of the table's content, we can conclude the following:

a) The LongMergent model has the lowest perplexity mean across all sequence lengths, demonstrating the best performance, indicating that the LongMergent model has high predictive accuracy and low uncertainty in generating audio data, also validating its stronger generalization capabilities;

b) As sequence length increases, the perplexity of some other models with poorer performance tends to increase, while the perplexity of the LongMergent model shows a decreasing trend, indicating that the LongMergent model has a high level of control over the harmony when integrating multiple short-text audio files.

Additionally, we invited several human listeners with professional music knowledge to subjectively evaluate the model's generated data. In the subjective evaluation, to measure the consistency of judgments among different raters, we use the Kappa coefficient. This statistical measure accounts for chance agreement and ranges from −1 to 1. Its formula is:

$$Kappa = \frac{P_0 - P_e}{1 - P_e} \tag{6}$$

where P_0 is the observed agreement, and P_e is the expected chance agreement. A Kappa value near 1 indicates high rater agreement, near 0 suggests agreement at chance level, and negative values mean agreement less than chance. Calculating the Kappa coefficient allows for a more accurate assessment of the reliability of subjective evaluation results.

We use the criteria outlined in Section 4.1. After tallying the scoring results, we will provide a general range of fluctuation. For detailed content, see **Table 6**.

Table 6. Results of subjective evaluation. ST and LT represent short-term and long-term structure, respectively. For all subjective indicators, the numerical display is the statistical mean ± standard deviation.

	Music Transformer	**Transformer-XL**	**Longformer**	**Museformer**	**LongMergent**
Musicality	6.00 ± 2.21	6.10 ± 2.19	6.46 ± 1.81	6.88 ± 1.95	6.55 ± 2.04
ST structure	6.90 ± 1.76	7.40 ± 1.81	7.60 ± 1.47	7.86 ± 1.51	7.62 ± 1.63
LT structure	5.30 ± 2.58	6.26 ± 2.78	6.18 ± 2.54	6.72 ± 2.74	6.77 ± 1.94
Overall	5.90 ± 1.90	6.44 ± 2.01	5.78 ± 2.64	7.12 ± 1.81	6.62 ± 2.06

We performed a Kappa test on the subjective evaluation results to assess the consistency among raters. The calculated *Kappa* value of 0.75 indicates substantial agreement. According to Landis and Koch's criteria, a Kappa value between 0.61 and 0.80 signifies "substantial agreement", suggesting our evaluation results are highly reliable.

The subjective evaluation results presented in **Table 6** show that LongMergent outperforms in all subjective assessment indicators, indicating that it has a higher capability in all aspects of music generation within the sensory range of human listeners. Specifically,

a) The LongMergent model scores slightly higher than other models in terms of musicality, suggesting that listeners perceive the music generated by the LongMergent model as more pleasing and authentic compared to music composed by humans;
b) In structure-related indicators, especially long-term structure, the LongMergent model has a distinct advantage, once again demonstrating the high performance of the LongMergent model in music fusion.

6. Conclusion

This paper introduces the LongMergent system, designed to address numerous challenges in the field of artificial intelligence-empowered music processing. By integrating a variety of music-related tools and autonomous workflows, the

LongMergent system provides users with a unified and efficient platform for the automated processing of music generation and understanding tasks, greatly simplifying the processes of music creation and analysis. The experimental results demonstrate that LongMergent is efficient and capable of generating music with good quality and structure.

However, the LongMergent system also has certain limitations. For instance, when dealing with certain complex music styles, such as classical music, the model struggles to fully capture their depth and complexity, and there is room for improvement in generating classical music pieces with high artistic expressiveness and intricate structures. This is a big challenge for professional musicians who need to create high-quality music works. To achieve the so-called "beautiful" level, it takes a lot of time and computing power to debug the model, and the work efficiency will be reduced. Moreover, the music generated by the LongMergent system is the result of multiple sub-tools working in concert; its overall performance is affected by the performance of these sub-tools. Should a singular available sub-tool have significant limitations, the final music generated will be impacted, and it cannot be compensated for by other tools. For the average music lover without professional music knowledge, this is likely to be an irreparable flaw. We believe that this issue can be mitigated by training large models on a vast scale of data. Lastly, we anticipate that the LongMergent model can adapt to more tasks and domains. Although its current functionalities are relatively comprehensive and can meet the basic generation needs of most users, researchers who specialize in this industry still need to spend a lot of economic cost to obtain higher computing power to meet their needs; there is still a long way to go to delve deeper into the professional field of music, requiring the supplementation of a multitude of high-performance tools.

Author contributions: Conceptualization, HL and ZL; methodology, HL; software, HL, KY and ZH; validation, HL, KY and CX; formal analysis, XL; investigation, HL and C; resources, ZL; data curation, HL; writing—original draft preparation, HL; writing—review and editing, HL and ZL; visualization, HL; supervision, ZL; project administration, HL; funding acquisition, ZL and C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Graduate Research and Practice Projects of Minzu University of China under Grant SJCX2024013, in part by the Fundamental Research for the Central Universities under Grant 2023QNYL24.

Conflict of interest: The authors declare no conflict of interest.

References

1. Gómez E, Gouyon F, Herrera P, Amatriain X. Using and enhancing the current MPEG-7 standard for a music content processing tool. Advances in Engineering Software. 2003.
2. Meng F, Zhang C, Liu N. Music style classification using deep convolutional neural networks. In: Proceedings of the 2020 3rd International Conference on Computer Graphics, Vision and Information Security (CGVIS). IEEE; 2020. pp. 87–91.
3. Hadjeres G, Pachet F, Nielsen F. DeepBach: A Steerable Model for Bach Chorales Generation. arXiv. 2016. doi: 10.48550/ARXIV.1612.01010

4. Chen J, Tan X, Luan J, et al. HiFiSinger: Towards High-Fidelity Neural Singing Voice Synthesis. arXiv. 2020. doi: 10.48550/ARXIV.2009.01776
5. Yu B, Lu P, Wang R, et al. Museformer: Transformer with fine- and coarse-grained attention for music generation. In: Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS); 28 November–8 December 2022.
6. Shen Y, Song K, Tan X, et al. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. arXiv. 2023.
7. Yu D, Song K, Lu P, et al. MusicAgent: An AI agent for music understanding and generation with Large Language Models. arXiv. 2023.
8. Chen Y, Huang L, Gou T. Applications and Advances of Artificial Intelligence in Music Generation: A Review. arXiv. 2024. doi: 10.48550/ARXIV.2409.03715
9. Agostinelli A, Denk TI, Borsos Z, et al. MusicLM: Generating music from text. arXiv. 2023.
10. Sun T, Zhang X, He Z, et al. MOSS: An Open Conversational Large Language Model. Machine Intelligence Research. 2024;21(5):888–905. DOI: 10.1007/s11633-024-1502-8.
11. Wang L, Kawakami K, van den Oord A. Contrastive Predictive Coding of Audio with an Adversary. Interspeech 2020. 2020; 826–830. doi: 10.21437/interspeech.2020-1891
12. Wu S, Yu D, Tan X, Sun M. CLaMP: Contrastive Language-Music Pre-training for Cross-Modal Symbolic Music Information Retrieval. arXiv. 2023. doi: 10.48550/ARXIV.2304.11029
13. Stöter F, Virtanen T. A Multichannel Nonnegative Matrix Factorization Approach to Sound Scene Analysis. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2016; 24(9): 1652–1663.
14. Engel J, Agrawal S, Chen D, et al. GANSynth: Adversarial Neural Audio Synthesis. In: Proceedings of the International Conference on Machine Learning (ICML); 10–15 June 2019; Long Beach, CA, USA.
15. Luo Y, Chen Z, Hershey JR, et al. Deep clustering and conventional networks for music separation: Stronger together. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2017; 61–65. doi: 10.1109/icassp.2017.7952118
16. Ji S, Yang X, Luo J. A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges. ACM Computing Surveys. 2023; 56(1): 1–39. doi: 10.1145/3597493
17. Min S, Lyu X, Holtzman A, et al. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? arXiv. 2022. doi: 10.48550/ARXIV.2202.12837
18. Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems. 2022; 35: 27730–27744.
19. Wu C, Yin S, Qi W, et al. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. arXiv. 2023. doi: 10.48550/ARXIV.2303.04671
20. Liu S, Hussain AS, Wu Q, et al. M2UGen: Multi-modal Music Understanding and Generation with the Power of Large Language Models. arXiv. 2023. doi: 10.48550/ARXIV.2311.11255
21. Chen C, Hu Y, Wang S, et al. Audio Large Language Models Can Be Descriptive Speech Quality Evaluators. arXiv. 2025. doi: 10.48550/ARXIV.2501.17202
22. Zeng G, Ding W, Xu B, et al. Adaptable and Precise: Enterprise-Scenario LLM Function-Calling Capability Training Pipeline. In: Proceedings of the 2025 International Conference on Learning Representations (ICLR); 24–28 April 2025.
23. Huang C-ZA, Vaswani A, Uszkoreit J, et al. Music transformer: Generating music with long-term structure. In: Proceedings of International Conference on Learning Representations (ICLR); 30 April–3 May 2018.
24. Dai Z, Yang Z, Yang Y, et al. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL); 28 July–2 August 2019; Florence, Italy. pp. 2978–2988.
25. Beltagy I, Peters ME, Cohan A. Longformer: The Long-Document Transformer. arXiv. 2020. doi: 10.48550/ARXIV.2004.05150.