MusicAgent: An AI Agent for Music Understanding and Generation with Large Language Models

Dingyao Yu[1,2], **Kaitao Song**[2], **Peiling Lu**[2], **Tianyu He**[2]
Xu Tan[2], **Wei Ye**[1*], **Shikun Zhang**[1*], **Jiang Bian**[2]
Peking University[1], Microsoft Research Asia[2]
{yudingyao, wye, zhangsk}@pku.edu.cn,
{kaitaosong, peil, tianyuhe, xuta, jiabia}@microsoft.com

https://github.com/microsoft/muzic

Abstract

AI-empowered music processing is a diverse field that encompasses dozens of tasks, ranging from generation tasks (e.g., timbre synthesis) to comprehension tasks (e.g., music classification). For developers and amateurs, it is very difficult to grasp all of these task to satisfy their requirements in music processing, especially considering the huge differences in the representations of music data and the model applicability across platforms among various tasks. Consequently, it is necessary to build a system to organize and integrate these tasks, and thus help practitioners to automatically analyze their demand and call suitable tools as solutions to fulfill their requirements. Inspired by the recent success of large language models (LLMs) in task automation, we develop a system, named *MusicAgent*, which integrates numerous music-related tools and an autonomous workflow to address user requirements. More specifically, we build 1) toolset that collects tools from diverse sources, including Hugging Face, GitHub, and Web API, etc. 2) an autonomous workflow empowered by LLMs (e.g., ChatGPT) to organize these tools and automatically decompose user requests into multiple sub-tasks and invoke corresponding music tools. The primary goal of this system is to free users from the intricacies of AI-music tools, enabling them to concentrate on the creative aspect. By granting users the freedom to effortlessly combine tools, the system offers a seamless and enriching music experience. The code is available on GitHub[1] along with a brief instructional video[2].

1 Introduction

AI-empowered music processing is a multifaceted and intricate domain, encompassing a diverse range of tasks. Mastering this field presents a challenging endeavor due to the wide array of tasks it involves. Generally, the realm of music includes various generation and comprehension tasks, such as songwriting (Sheng et al., 2021; Ju et al., 2021), music generation (Agostinelli et al., 2023; Dai et al., 2021; Lu et al., 2023; Lv et al., 2023), audio transcription (Benetos et al., 2018; Foscarin et al., 2020), music retrieval (Wu et al., 2023b), etc. Specifically, music is a complex art form that weaves together a variety of distinct elements, such as chords and rhythm, to create vibrant and intricate content. Previous works have frequently encountered challenges in collaborating to complete complex music tasks, primarily due to differences in music feature designs and variations across platforms. Therefore,

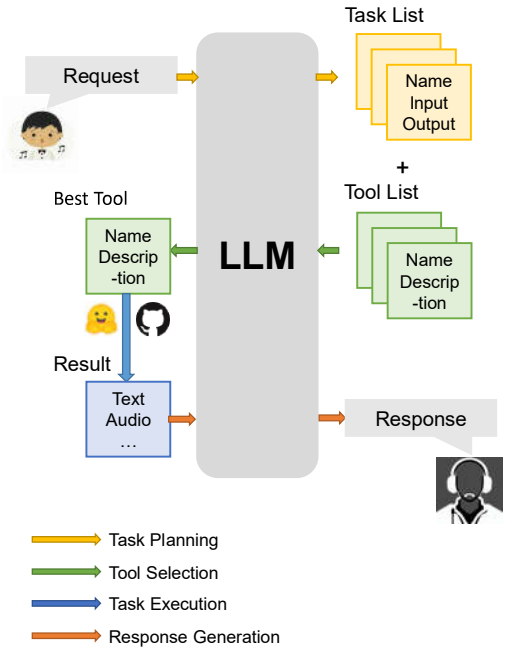

Figure 1: MusicAgent has gathered a rich collection of music-related tasks and diverse sources of tools, effectively integrating them with LLMs to achieve proficiency in handling complex music tasks.

*Corresponding Author: Wei Ye, wye@pku.edu.cn; Shikun Zhang, zhangsk@pku.edu.cn

[1] https://github.com/microsoft/muzic/tree/main/agent

[2] https://youtu.be/tpNynjdcBqA

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 246–255
December 6-10, 2023

how to build a system to automatically accomplish music-related tasks from the requests of users with varying levels of expertise is still an enticing direction worth exploring.

Recently, large language models (LLMs) have attracted considerable attention due to their outstanding performance in solving natural language processing (NLP) tasks (Brown et al., 2020; Ouyang et al., 2022; Zhang et al., 2022b; Chowdhery et al., 2022; Zeng et al., 2022; Touvron et al., 2023). The huge potentials of LLMs also inspire and directly facilitate many emerging techniques (e.g., in-context learning (Xie et al., 2021; Min et al., 2022), instruct tuning (Longpre et al., 2023; Wang et al., 2022), and chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022)), which also further elevate the capability of LLMs. On the basis of these LLM capabilities, many researchers have extended the scope of LLMs to various topics. They borrow the idea of acting LLMs as the controllers to orchestrate various domain-specific expert models for solving complex AI tasks, such as HuggingGPT (Shen et al., 2023), AutoGPT and other modality-specifics ones (Chen et al., 2022; Wu et al., 2023a; Huang et al., 2023). These successes also motivate us to explore the possibility to develop a system capable of assisting with various music-related tasks.

Distinguishing from other modalities, incorporating LLMs with music presents the following features and challenges:

1. **Tool Diversity**: On one hand, music-related tasks exhibit a wide range of diversity, and on the other hand, the corresponding tools for these tasks might not always reside on the same platform. These tools could be parameterized models available in open-source communities like GitHub, presented as software and applications, or even hosted through Web APIs for certain retrieval tasks. Considering all these factors is crucial when undertaking a comprehensive music workflow.

2. **Cooperation**: The collaboration between music tools is also constrained by two factors. First, the diversity of music domain tasks leads to the absence of explicit input-output modality standards. Second, even when the modalities are identical, the music formats may differ, for instance, between symbolic music and audio music.

To address these issues, we introduce MusicAgent, a specialized system designed to tackle the challenges. Inspired by recent work like HuggingGPT (Shen et al., 2023), MusicAgent is a framework that utilizes the power of LLMs as the controller and massive expert tools to accomplish user instructions, just as illustrated in Figure 1. For the toolset, in addition to utilizing the models provided by Hugging Face, we further integrate various methods from different sources, including code from GitHub and Web APIs. To make collaboration between diverse tools, MusicAgent enforces standardized input-output formats across various tasks to promote seamless cooperation between tools. As a music-related system, all samples are trimmed to fit within a single audio segment, facilitating fundamental music operations among samples. For more system details and guidance on integrating additional tools, please refer to Section 3.

Overall, the MusicAgent presents several significant contributions:

- **Accessibility:** MusicAgent eliminates the need to master complex AI music tools. By utilizing LLMs as the task planner, the system dynamically selects the most suitable methods for each music-related task, making music processing accessible to a broader audience.

- **Unity:** MusicAgent bridges the gap between tools from diverse sources by unifying the data format (e.g., text, MIDI, ABC notation, audio). The system enables seamless cooperation among tools on different platforms.

- **Modularity:** MusicAgent is highly extensible, allowing users to easily expand its functionality by implementing new functions, integrating GitHub projects, and incorporating Hugging Face models.

2 Related Works

2.1 AI-Empowered Music Processing

Music generation and understanding are multi-faceted tasks that encompass various sub-tasks. In the realm of music generation, these tasks involve melody generation (Yu et al., 2020; Zhang et al., 2022a; Yu et al., 2022), audio generation (Donahue et al., 2018), singing voice synthesis (Ren et al., 2020; Lu et al., 2020), and sound mixing. In contrast, music understanding encompasses track separation (Défossez et al., 2019), audio recognition, score transcription (Bittner et al., 2022), audio

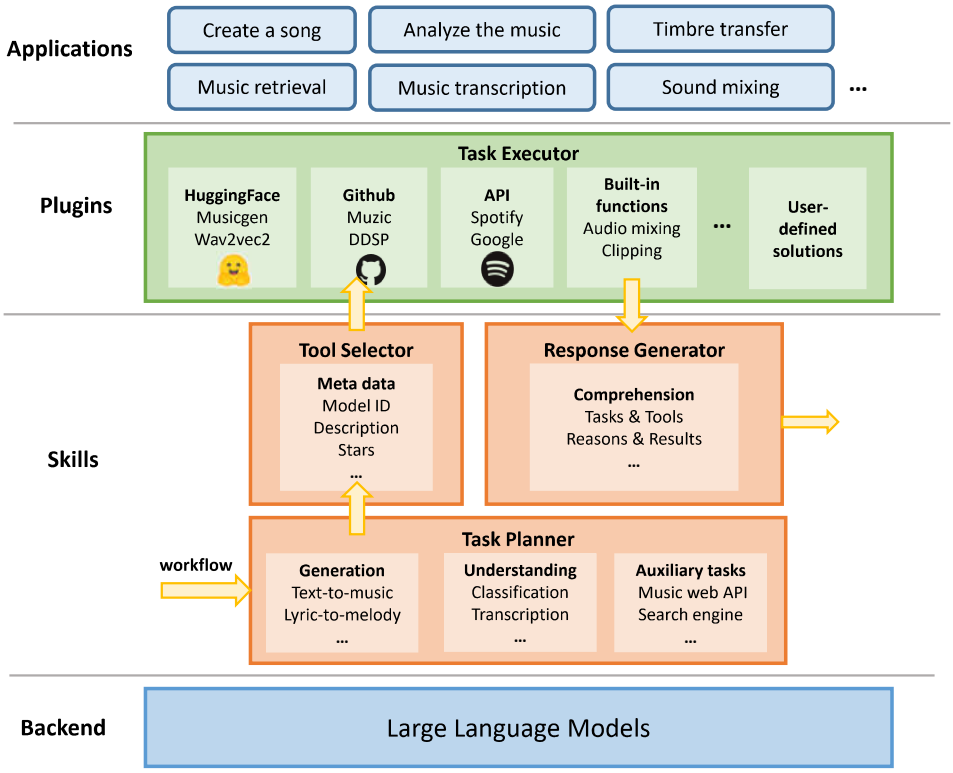

Figure 2: MusicAgent consists of four core components: the task planner, tool selector, task executor, and response generator. Among these, the task planner, tool selector, and response generator are built upon language language models (LLMs). When users make requests, MusicAgent decomposes and organizes the requests into subtasks. The system then selects the most suitable tool for each task. The chosen tool processes the input and populates the anticipated output. The LLM subsequently organizes the output, culminating in a comprehensive and efficient music processing system.

classification (Choi et al., 2017; Zeng et al., 2021), and music retrieval (Wu et al., 2023b). In addition to these diverse and complex music-related tasks, another significant challenge in traditional music processing is substantial differences in input and output formats across each task. These diversities in tasks and data formats also hinder the unification in music processing, which makes it difficult for us to develop a copilot for solving different musical tasks. Therefore, in this paper, we will discuss how to design a copilot to unified musical data format and combine these tools to automatically accomplish tasks by utilizing large language model.

2.2 Large Language Models

The field of natural language processing (NLP) is undergoing a revolutionary shift due to the emergence of large language models (LLMs). These models (Brown et al., 2020; Touvron et al., 2023) have exhibited powerful performance in various language tasks, such as translation, dialogue modeling, and code completion, making them a focal point in NLP.

Based on these advantages, LLMs have been applied to many applications. Recently, a new trend is to use LLMs to build autonomous agents for task automation, just like AutoGPT [3] and HuggingGPT (Shen et al., 2023). In these works, they will leverage an LLM as the controller to automatically analyze user requests and then invoke the appropriate tool for solving tasks. Although there are some successful trials in vision (Chen et al., 2022) or speech (Huang et al., 2023), it is still challenging to build an autonomous agent for music processing, due to its diversity and complexity in tasks and data. Therefore, we present a system called MusicAgent, which integrates various functions to handle multiple music-related tasks, to accomplish requests from different users, including novices and professionals.

[3] https://github.com/Significant-Gravitas/Auto-GPT

Table 1: Overview of tasks and the associated example tools in MusicAgent.

Task	Input	Output	Task Type	Example Tool
text-to-symbolic-music	text	symbolic music	Generation	MuseCoco[4]
lyric-to-melody	text	symbolic music	Generation	ROC[5]
singing-voice-synthesis	text	audio	Generation	HiFiSinger[6]
text-to-audio	text	audio	Generation	AudioLDM
timbre-transfer	audio	audio	Generation	DDSP[7]
accompaniment	symbolic music	symbolic music	Generation	GetMusic[8]
music-classification	audio	text	Understanding	Wav2vec2
music-separation	audio	audio	Understanding	Demucs
lyric-recognition	audio	text	Understanding	Whisper-large-zh[9]
score-transcription	audio	text	Understanding	Basic-pitch
artist/track-search	text	audio	Auxiliary	Spotify API[10]
lyric-generation	text	text	Auxiliary	ChatGPT
web-search	text	text	Auxiliary	Google API

3 MusicAgent

MusicAgent is a comprehensive system that enhances the capabilities of large language models (LLMs) and tailors them to the music domain by integrating additional data sources, dependent tools, and task specialization. As illustrated in Figure 2, MusicAgent designs an LLM-empowered autonomous workflow, which includes three key skills: Task Planner, Tool Selector, and Response Generator. These skills, along with the music-related tools forming the Task Executor, are integrated, resulting in a versatile system capable of executing various applications. In this section, we will delve into different aspects of this system, exploring its functionalities and contributions to the field of music processing.

3.1 Tasks and Tools Collection

Table 1 provides a comprehensive overview of the music-related tasks and representative tools gathered in the current MusicAgent. We have organized the task sets based on the music processing flow illustrated in Figure 3. Aside from generation and understanding tasks, the collected tasks are primarily categorized into three groups:

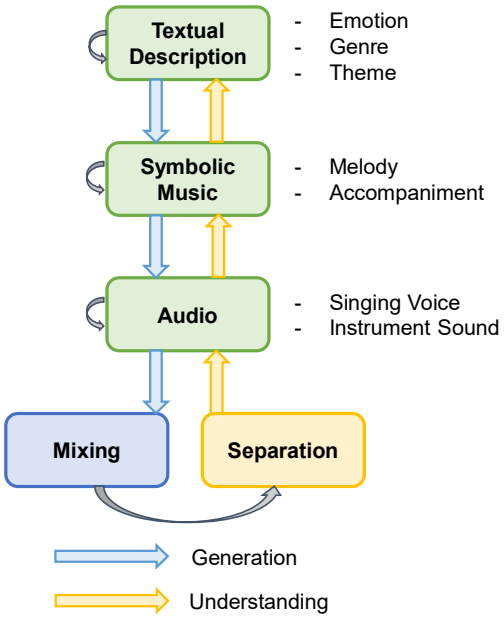

Figure 3: MusicAgent collects tasks and tools within the framework of music generation and understanding. It encompasses various tasks, including single-modal tasks and modality transfer tasks, such as converting sheet music to audio through singing voice synthesis.

Generation tasks: This category includes *text-to-music*, *lyric-to-melody*, *singing-voice-synthesis*, *timbre-transfer*, *accompaniment*, and etc. These tasks enable the collaborative music generation starting from simple descriptions.

Understanding tasks: The tasks of *music-classification*, *music-separation*, *lyric recognition*,

[4] https://github.com/microsoft/muzic/tree/main/musecoco
[5] https://github.com/microsoft/muzic
[6] https://github.com/CODEJIN/HiFiSinger
[7] https://github.com/magenta/ddsp
[8] https://github.com/microsoft/muzic/tree/main/musecoco/getmusic
[9] https://huggingface.co/jonatasgrosman/whisper-large-zh-cv11
[10] https://spotify.com

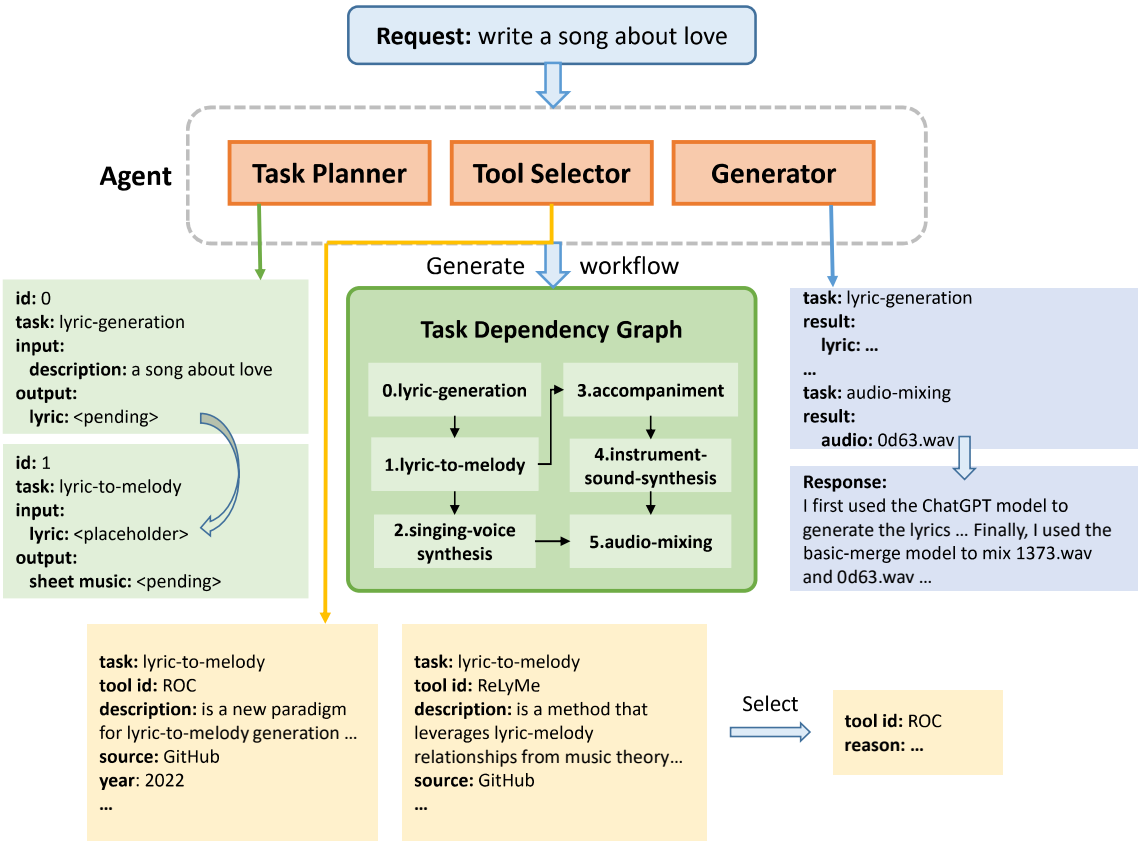

Figure 4: The LLM backend is responsible for the following steps: The Task Planner takes user requests and produces parsed task queue, the Tool Selector chooses suitable tools, and the Response Generator collects tool outputs and organizes the responses.

and *music-transcription* are under this category. Combining these tasks enables the conversion of music into symbolic representation and the analysis of various music features.

Auxiliary tasks: This category encompasses web search and various audio processing toolkits. Web search includes text search using the *Google API*, as well as music search through the *Spotify API*. These tasks primarily provide rich data sources and perform basic operations on audio/MIDI/text data, serving as auxiliary functions.

Furthermore, Figure 3 illustrates the utilization of three main data formats in the system: i) text, which includes lyric, genre or any other attributes related to the music. ii) sheet music, represented as MIDI files, describes the score of the music. iii) audio, containing the sound of the music.

3.2 Autonomous Workflow

The MusicAgent system consists of two parts: the autonomous workflow and the plugins. The autonomous workflow serves as the core LLM interaction component, as shown in Figure 2, and it comprises three skills: Task Planner, Tool Selector, and Response Generator, all supported by the LLM. Figure 4 further demonstrates how these components work together harmoniously.

Task Planner: The Task Planner plays a critical role in converting user instructions into structured information, as most existing music tools only accept specialized inputs. The user input processed by the Task Planner will form the backbone of the entire workflow, encompassing the determination of each subtask and its corresponding input-output format, as well as the dependencies between the subtasks, creating a dependency graph. Leveraging in-context learning, MusicAgent demonstrates excellent task decomposition performance. We provide task planner descriptions, supported tasks, and information structure in the prompt, along with several examples of music task-related decompositions. The user's interaction history and current

input will replace the content at the corresponding position in the prompt. By utilizing the Semantic Kernel (Microsoft, 2023), users can insert the required task flow in text format, thereby enhancing task planning effectiveness.

Tool Selector: The Tool Selector chooses the most appropriate tool from the open-source tools relevant to a specific subtask. Each tool is associated with its unique attributes, such as textual descriptions, download count, star ratings, and more. By incorporating these tool attributes with the user input, LLM presents the tool's ID and corresponding reasoning for what it deems the most suitable selection. Users have the flexibility to adjust the tool attributes and determine how LLM interprets these attributes. For instance, users can emphasize download count to meet diverse requirements.

Response Generator: The Response Generator gathers all intermediate results from the execution of subtasks and ultimately compiles them into a coherent response. Examples in Figure 5 demonstrate how LLM organizes the tasks and results to generate answers.

3.3 Plugins

When all the dependent tasks of a subtask have been completed, and all inputs have been instantiated, the LLM backend passes the task to the Task Executor, where the tool selects the necessary parameters from the inputs. Additionally, the tool needs to identify the task type, as a tool may handle multiple tasks.

MusicAgent stores model parameters on the CPU and only loads them into the GPU when actively in use. This approach is especially advantageous for users with limited GPU memory, as it optimizes resource utilization and ensures smooth task execution without overburdening the GPU memory.

4 System Usage

In this section, we will provide comprehensive guidelines on how to effectively use the MusicAgent toolkit.

4.1 Code Usage

Users have the flexibility to run this system either by following the instructions on GitHub or by integrating it as a module in their code or using it through the command line for more advanced usage, enabling the incorporation of custom tools. As depicted in Listing 1, users can add custom task types, update tool attributes, and design prompts for each subtask, enhancing support for specific tasks. It is important to note that embedding the prompt in the history is a temporary operation, and there is a possibility of overlap if the context exceeds the limit. For permanent storage, it is recommended to directly include the prompt in the code.

```
## 1. Initialze the agent
from agent import MusicAgent
music_agent = MusicAgent(CONFIG_PATH)

## 2. Add custom tasks and tools
music_agent.task_map[MY_TASK].append(MY_TOOL)
music_agent.pipelines.append(MY_TOOL_CLASS)
# Update prompts
music_agent._init_task_context()
music_agent._init_tool_context()

## 3. Update tool's information
music_agent.update_tool_attributes(
    MY_TOOL, {"stars":..,"likes":..})
music_agent._init_tool_context()

## 4. Update the prompt
# Take task planner as an example
# There is a risk of being overwritten
music_agent.task_context["history"] +=
    "MY CUSTOM PROMPT"

## 5. Chat with the agent
music_agent.chat("Generate a song...")
```

Listing 1: Code usage of MusicAgent

4.2 Demo Usage

Apart from command-line usage, we have also provided a Gradio demo for users, where an OpenAI token is required. In the Gradio demo, users can directly upload audio and visually observe all the intermediate results generated by the system, as depicted in Figure 6. Additionally, although MusicAgent includes built-in context truncation, users can still clear all LLM interaction history in the interface to refresh the agent.

5 Conclusion

In this paper, we introduce MusicAgent, an LLM-powered autonomous agent in the music domain. Our system can be considered as an auxiliary tool to help developers or audiences to automatically analyze user requests and select appropriate tools as solutions. Moreover, our framework directly integrates numerous music-related tools from various sources (e.g., Hugging Face, GitHub, Web search and etc). We also adapt the autonomous

workflow to enable better compatibility in musical tasks and allow users to extend its toolset. In the future, we also further envision integrating more music-related functions into MusicAgent.

Acknowledgements

We extend our gratitude to all anonymous reviewers and members of the Machine Learning group at Microsoft Research Asia for their valuable contributions and insightful suggestions in the development of this system.

References

Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*.

Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. 2018. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30.

Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. 2022. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. 2022. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 2392–2396. IEEE.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Shuqi Dai, Zeyu Jin, Celso Gomes, and Roger B Dannenberg. 2021. Controllable deep melody generation via hierarchical music structure representation. *arXiv preprint arXiv:2109.00663*.

Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. 2019. Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*.

Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.

Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. 2020. Asap: a dataset of aligned scores and performances for piano transcription. In *International Society for Music Information Retrieval Conference*, CONF, pages 534–541.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 2023. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*.

Zeqian Ju, Peiling Lu, Xu Tan, Rui Wang, Chen Zhang, Songruoyao Wu, Kejun Zhang, Xiangyang Li, Tao Qin, and Tie-Yan Liu. 2021. Telemelody: Lyric-to-melody generation with a template-based two-stage method. *arXiv preprint arXiv:2109.09617*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. 2020. Xiaoicesing: A high-quality and integrated singing voice synthesis system. *arXiv preprint arXiv:2006.06261*.

Peiling Lu, Xin Xu, Chenfei Kang, Botao Yu, Chengyi Xing, Xu Tan, and Jiang Bian. 2023. Musecoco: Generating symbolic music from text. *arXiv preprint arXiv:2306.00110*.

Ang Lv, Xu Tan, Peiling Lu, Wei Ye, Shikun Zhang, Jiang Bian, and Rui Yan. 2023. Getmusic: Generating any music tracks with a unified representation and diffusion framework. *arXiv preprint arXiv:2305.10841*.

Microsoft. 2023. Semantic kernel. https://github.com/microsoft/semantic-kernel.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu. 2020. Deepsinger: Singing voice synthesis with data mined from the web. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1979–1989.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2021. Songmass: Automatic song writing with pre-training and alignment constraint. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13798–13805.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Shangda Wu, Dingyao Yu, Xu Tan, and Maosong Sun. 2023b. Clamp: Contrastive language-music pretraining for cross-modal symbolic music information retrieval. *arXiv preprint arXiv:2304.11029*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. 2022. Museformer: Transformer with fine-and coarse-grained attention for music generation. *Advances in Neural Information Processing Systems*, 35:1376–1388.

Yi Yu, Florian Harscoët, Simon Canales, Gurunath Reddy M, Suhua Tang, and Junjun Jiang. 2020. Lyrics-conditioned neural melody generation. In *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II 26*, pages 709–714. Springer.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. 2021. Musicbert: Symbolic music understanding with large-scale pre-training. *arXiv preprint arXiv:2106.05630*.

Chen Zhang, Luchin Chang, Songruoyao Wu, Xu Tan, Tao Qin, Tie-Yan Liu, and Kejun Zhang. 2022a. Relyme: Improving lyric-to-melody generation by incorporating lyric-melody relationships. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1047–1056.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Appendix

Figure 5: Sample Q&A for users and agents.

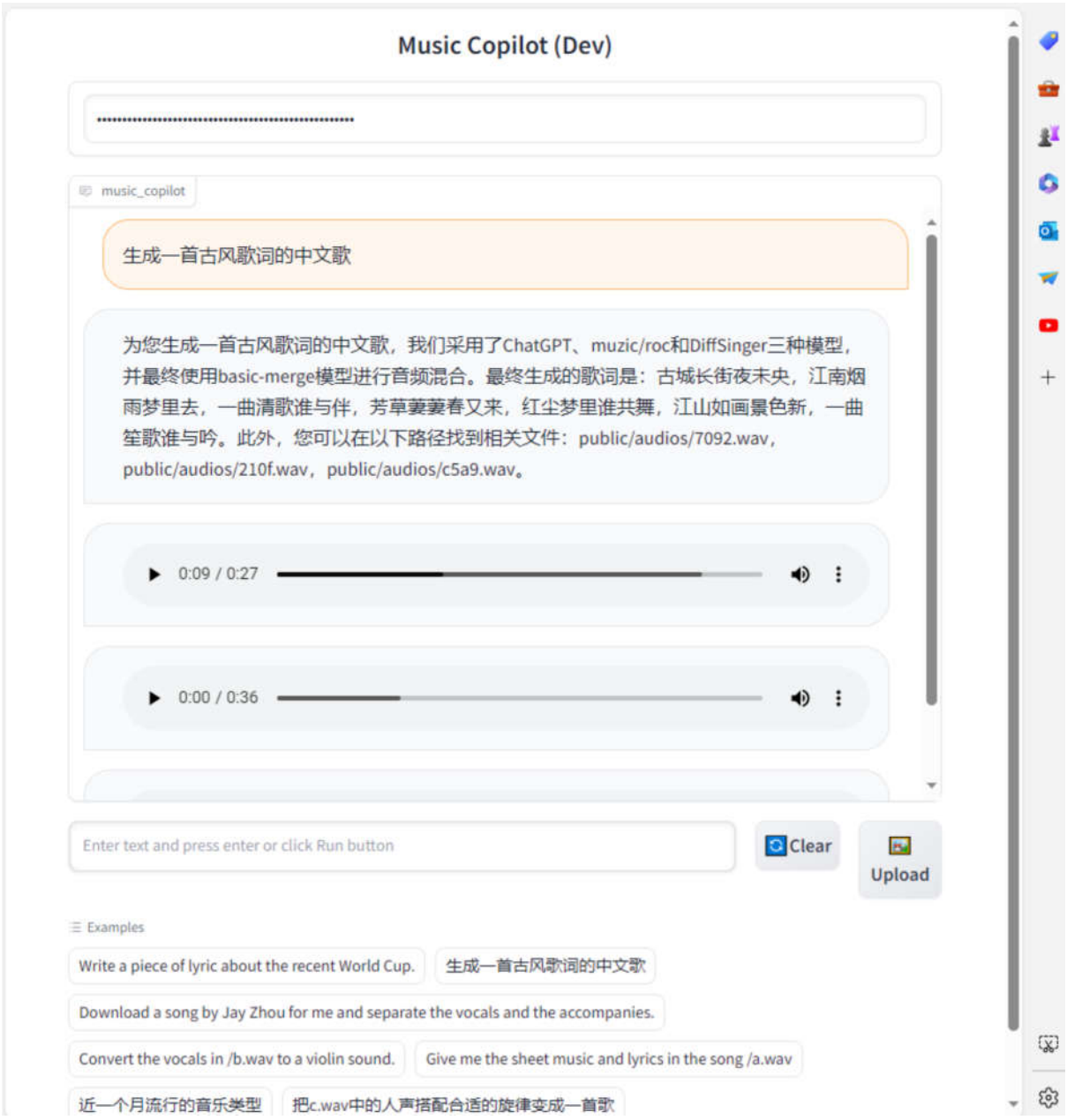

Figure 6: Gradio Demomstration.