TalkPlay-Tools: Conversational Music Recommendation with LLM Tool Calling

Seungheon Doh [1,2,*], **Keunwoo Choi** [1,2,*], **Juhan Nam** [1]
[1] KAIST, South Korea, [2] talkpl.ai, USA
https://talkpl.ai/talkplay_tools

Abstract

While the recent developments in large language models (LLMs) have successfully enabled generative recommenders with natural language interactions, their recommendation behavior is limited, leaving other simpler yet crucial components such as metadata or attribute filtering underutilized in the system. We propose an LLM-based music recommendation system with tool calling to serve as a unified retrieval-reranking pipeline. Our system positions an LLM as an end-to-end recommendation system that interprets user intent, plans tool invocations, and orchestrates specialized components—boolean filters (SQL), sparse retrieval (BM25), dense retrieval (embedding similarity), and generative retrieval (semantic IDs). Through tool planning, the system predicts which types of tools to use, their execution order, and the arguments needed to find music matching user preferences, supporting diverse modalities while seamlessly integrating multiple database filtering methods. We demonstrate that this unified tool-calling framework achieves competitive performance across diverse recommendation scenarios by selectively employing appropriate retrieval methods based on user queries, envisioning a new paradigm for conversational music recommendation systems.

1 Introduction

Music recommendation has long been shaped by how to filter databases according to user preferences. Early systems relied on boolean filtering [1] over listening history logs and metadata such as title, artist, and release year. The advent of large-scale music item catalogs and user-item feature learning algorithms [11] shifted the field toward retrieval and reranking pipelines driven by large-scale user and item embeddings [23, 8]. With advances in representation learning, the field has progressed beyond listening history to develop deeper content understanding [26, 16]. Moreover, music representations for audio [12], lyrics [18], and visual artwork [7] has enabled example-based recommendation, while multimodal music representations [13, 9, 6, 5, 29, 28, 15, 24] has supported natural language query understanding. Recent advances in large language model (LLM)-based recommendation systems [4, 17] demonstrate that items can be represented in a semantic space and quantized into discrete Semantic IDs [19, 14], enabling both generative recommendation and interactive dialogue with users. In these methods, users are allowed to engage actively to discover music that matches their goals and preferences through multi-turn interactions.

However, relying on a single retrieval method has clear limitations in identifying items that satisfy user needs [27]. Production-level recommenders [3, 10] operate based on multiple stages and routed retrieval–reranking pipelines and must strictly satisfy operational constraints (e.g., user profile, genre, mood, activity, newness, etc.), while simultaneously reflecting a listener's history and text queries. Without combining multiple types of retrieval methods, it cannot fully capture all the relevant context of recommendations.

[*]Equal contribution. Corresponding authors: seungheondoh@kaist.ac.kr, juhan.nam@kaist.ac.kr;

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Music.

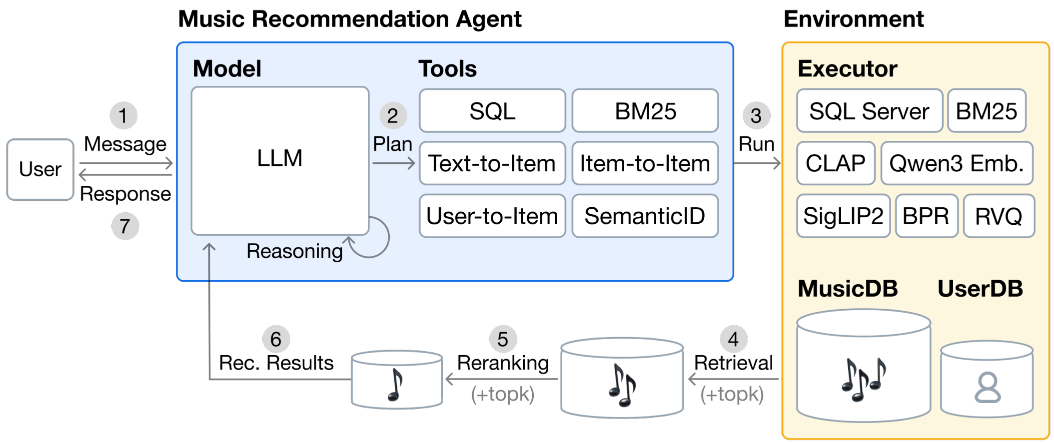

Figure 1: Overview of Music Recommendation Agents with Tool Calling.

In this paper, we present a conversational music recommender with tool-calling. Our contributions are: (i) formulating conversational music recommendation with multiple retrieval methods, enabling the LLM to predict tool types, order, and arguments conditioned on a user profile and a dialogue context; (ii) designing a unified tool set that composes of boolean filtering (SQL), sparse retrieval (BM25), dense retrieval across modalities (text-to-item, item-to-item, and user-to-item), and generative retrieval (Semantic IDs) under a single agent; and (iii) demonstrating zero-shot effectiveness on a conversational recommendation benchmark with improved Hit@K over strong baselines, together with detailed analyses of their behaviors.

2 Music Recommendation with Tool Calling

Figure 1 illustrates our framework that consists of two main components. The first component is the *Music Recommendation Agent*, comprising the LLM and tools. The second component is the *External Environment* that executes the tools and performs the final recommendation through retrieval and reranking.

2.1 Problem Formulation

Given a user u with profile p_u, a previous conversation state s_{t-1}, and the user query q_t at the current turn t, the proposed system produces a ranked list of music items m_t by (i) generating tool calls with a LLM and (ii) filtering database $\mathcal{D}$ using the predicted tools $\mathcal{C}_t$. Finally, the LLM is called again to generate a natural language response r_t that provides a conversational explanation of the recommendations while maintaining dialogue context.

$$\mathcal{C}_t = \text{LLM}(q_t, s_{t-1}, p_u; \mathcal{P}_{tool}, \mathcal{T}) \tag{1}$$

$$m_t = \text{ToolEnv}(\mathcal{C}_t; \mathcal{D}) \tag{2}$$

$$r_t = \text{LLM}(m_t, q_t, s_{t-1}, p_u; \mathcal{P}_{response}) \tag{3}$$

Let $\mathcal{T}$ denote the available tools, $\mathcal{P}$ the prompt, and $\mathcal{C}_t = \left[(\text{tool}_n, \text{args}_n)\right]_{n=1}^{N}$ the LLM-predicted sequence of tool calls with the number of tools N. The tool environment executes each and every $(\text{tool}_n, \text{args}_n)$ pair to retrieve and rerank results into a final ranked list m_t.

The tool execution environment operates as a sequential pipeline where each tool's output directly influences the subsequent tool's input space, i.e., each tool filters and refines the track pool for downstream tools. Therefore, the order of the tools significantly affects the final recommendation quality. Through the prompt $\mathcal{P}_{tool}$, we guide the LLM to perform recommendations not as a single operation, but as a staged process consisting of retrieval and reranking phases, ensuring that the tool execution follows a sequential pipeline.

Table 1: Comparison of capabilities across different tools. Gray text indicates tools activated by in-context information rather than direct natural language user queries.

Tools	Capabilities	Environments	Query Examples
Boolean Retrieval			
SQL	Numeric Filtering	SQL Server	“Recent songs over 130 BPM"
Sparse Retrieval			
BM25	Lexical Matching	BM25 Index	“Songs from Adele’s 21"
Dense Retrieval			
Text-to-Item	Semantic Matching	Qwen3, CLAP, SigLIP2	“Play a calm piano piece"
Item-to-Item	Multimodal Matching	CLAP, SigLIP2, BPR	“Ok, more similar voices"
User-to-Item	Personalization	BPR	user_id:10021
Generative Retrieval			
Semantic ID	Multimodal In-Context	Residual VQ Tables	audio:semantic_id:[63, 36, 44, 3]

2.2 Tools

Our framework incorporates diverse tools that enable precise and flexible music recommendation through structured filtering and semantic retrieval approaches. Table 1 summarizes their capabilities, environments, and usage examples.

SQL (Boolean Retrieval): The SQL tool enables precise and structured queries on relational music metadata. We construct a single table with fields including title, artist, album, release date, tempo, key, and popularity. Tool arguments include query for SQL statements and topk for result limits. e.g., `sql(query="SELECT * FROM tracks WHERE date>=2020 ORDER BY tempo", topk=100)`.

BM25 (Sparse Retrieval): This tool provides classic token-based text retrieval using the BM25 ranking function [21]. BM25 offers lexical matching, making it particularly effective for text queries where typos are common and exact string matching is difficult. We construct five text corpora: title, artist, album, lyrics, and attributes (semantic tags). Tool arguments include query, corpus, and topk, e.g., `bm25(query="taylor swift songs", corpus="artist", topk=100)`.

Text-to-Item (Dense Retrieval): This tool enables semantic music discovery through natural language descriptions by mapping text queries to musical content across multiple modalities. Utilizing pretrained text [30] and multimodal encoders [29, 25], the tool retrieves the most similar items based on cosine similarity in the corresponding embedding space. Tool arguments include query, item modality type, vector database type, and topk, e.g., `text_to_item(query="Tracks with an album cover that shows a baby swimming underwater.", item_modality="image", vector_db="image", topk=20)`.

Item-to-Item (Dense Retrieval): This tool supports example-based recommendation by finding similar items using dense representations. The LLM generates track IDs based on previously recommended tracks or user-provided examples in multi-turn conversations, then we look up the corresponding embeddings [29, 25, 20] from pre-extracted vector databases to perform similarity-based retrieval. Tool arguments include track ID, item modality type, vector database type, and topk, e.g., `item_to_item(track_id="22L7bfCiAkJo5xGSQgmiIO", item_modality="audio", vector_db="audio", topk=20)`.

User-to-Item (Dense Retrieval): This tool provides personalized recommendations using user embeddings trained through listening history [20]. Unlike other tools activated by user queries, this tool is activated using the user ID from the user profile information. Tool arguments include user_Id and topk, e.g., `user_to_item(user_id=10021, topk=200)`.

Semantic IDs (Generative Retrieval): Semantic IDs are discrete representations derived from item content features [19, 14]. Content is encoded into dense embeddings, then quantized into discrete codebook indices using Residual Vector Quantizer (RVQ). Semantic IDs serve as in-context information to help the LLM understand the multimodal properties of music. We build inverted indexes from code positions to item IDs for fast lookup by exact code match or small edit distance. Tool arguments include item modality type, Semantic ID indices, and topk, e.g., `semantic_id(item_modality="audio", indices=[52, 42, 5, 9], topk=20)`.

Table 2: Conversational music recommendation results. QU, RG, TC indicate query understanding, response generation, tool calling capabilities, respectively.

Models	Methods	QU	RG	TC	Hit@1(↑)	Hit@10(↑)	Hit@20(↑)
BM25	Sparse	✓	✗	✗	0.017	0.073	0.107
Qwen3-LM + BM25	Generative	✓	✓	✗	0.018	0.076	0.110
Qwen3-LM + Tool (**Ours**)	Generative	✓	✓	✓	0.022	0.082	0.111

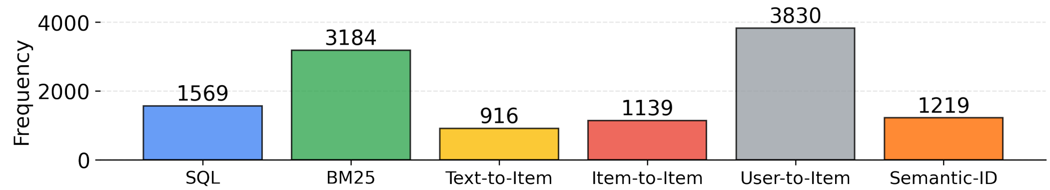

Figure 2: Tool Calling Frequency at First Attempt.

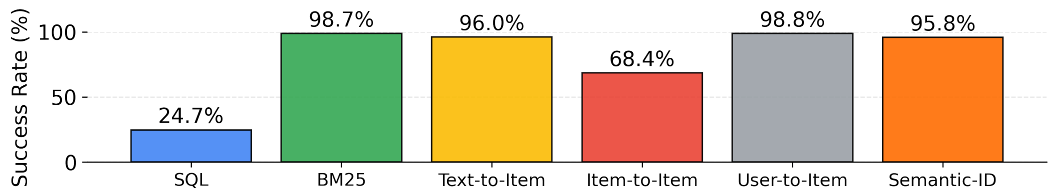

Figure 3: Tool Calling Success Rate at First Attempt.

3 Experiments

We use TalkPlayData 2 [2], a synthetic dataset designed for multimodal conversational music recommendation. The dataset contains diverse user profiles with demographic information (gender, age, country) from LFM-2b [22]. It also provides comprehensive multimodal music representations including listening history, metadata, semantic tags, lyrics, album art, and audio. We evaluate different conversational recommendation systems on 1000 test conversations, each consisting of 8 turns. For conversational recommendation, we measure Hit@K as our primary metrics with k={1,10,20}. We compare our tool-calling approach against several baseline methods: (1) BM25, a classic sparse retrieval method; (2) Qwen3-LM + BM25, which combines language model generation with fuzzy matching; and (3) our proposed Qwen3-LM + Tool Calling approach that leverages multiple tools. We use Qwen3-LM-4B [30] as our base model, [2] leveraging its strong reasoning capabilities for complex tool calling and query decomposition. We set the generation parameters to temperature=0.6 and top_p=0.95. We represent all identifiers (User IDs, Track IDs, Semantic IDs) as natural language strings rather than special tokens with vocabulary expansion.

4 Discussion

Our tool-calling based conversational recommender outperforms prior methods in zero-shot settings. As shown in Table 2, Hit@1 improves over Qwen3-LM + BM25 (0.022 vs 0.018), underscoring the effectiveness of multi-tool retrieval-reranking framework. When an individual tool call fails during inference, the system automatically retries, ensuring the pipeline completes. The first-attempt analysis in Figures 2 and 3 further shows higher invocation frequencies for natural language-friendly tools (SQL, BM25), while item-specific modalities (item-to-item, Semantic ID) are used less often. This likely reflects pretraining exposure: universal operators like SQL/BM25 are common in IR corpora, whereas item-to-item and Semantic ID tools have domain-specific function names and interfaces, making them less familiar to pretrained LLMs. In terms of success rates, more complex tools present challenges. SQL queries achieve only 27.4% due to syntactic complexity, invalid column names, and metadata–query mismatches caused by synonyms or typos. Item-to-item reaches 68.4%, reflecting the difficulty of predicting complete track IDs.

As future work, we will improve domain-specific tool calling reliability. Domain-specific tool calling requires reinforcement learning to optimize tool selection and execution strategies, which we plan to incorporate through tool-specific instruction tuning and RL-based policy learning to reduce retries and raise success rates.

[2] https://huggingface.co/Qwen/Qwen3-4B

References

[1] O. Celma. Music recommendation: In music recommendation and discovery, 2010.

[2] K. Choi, S. Doh, and J. Nam. Talkplaydata 2: An agentic synthetic data pipeline for multimodal conversational music recommendation. *arXiv preprint arXiv:2509.09685*, 2025.

[3] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[4] S. Doh, K. Choi, and J. Nam. Talkplay: Multimodal music recommendation with large language models. *arXiv preprint arXiv:2502.13713*, 2025.

[5] S. Doh, M. Lee, D. Jeong, and J. Nam. Enriching music descriptions with a finetuned-LLM and metadata for text-to-music retrieval. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[6] S. Doh, M. Won, K. Choi, and J. Nam. Toward universal text-to-music retrieval. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[7] C.-C. Hsia, K.-H. Lai, Y. Chen, C.-J. Wang, and M.-F. Tsai. Representation learning for image-based music recommendation. *arXiv preprint arXiv:1808.09198*, 2018.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *International conference on data mining (ICDM)*, 2008.

[9] Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. Ellis. Mulan: A joint embedding of music audio and natural language. *International Society for Music Information Retrieval (ISMIR)*, 2022.

[10] K. Jacobson, V. Murali, E. Newett, B. Whitman, and R. Yon. Music personalization at spotify. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 373–373, 2016.

[11] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[12] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam. Disentangled multidimensional metric learning for music similarity. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[13] I. Manco, E. Benetos, E. Quinton, and G. Fazekas. Contrastive audio-language learning for music. *International Society for Music Information Retrieval (ISMIR)*, 2022.

[14] M. J. Mei, F. Henkel, S. E. Sandberg, O. Bembom, and A. F. Ehmann. Semantic ids for music recommendation. *arXiv preprint arXiv:2507.18800*, 2025.

[15] S. Oramas, A. Ferraro, A. Sarasua, and F. Gouyon. Talking to your recs: Multimodal embeddings for recommendation and retrieval. In *MuRS 2024: 2nd Music Recommender Systems Workshop*, 2024.

[16] S. Oramas, O. Nieto, M. Sordo, and X. Serra. A deep multimodal approach for cold-start music recommendation. In *Proceedings of the 2nd workshop on deep learning for recommender systems*, pages 32–37, 2017.

[17] E. Palumbo, G. Penha, A. Damianou, J. L. R. García, T. C. Heath, A. Wang, H. Bouchard, and M. Lalmas. Text2tracks: Prompt-based music recommendation via generative retrieval. *arXiv preprint arXiv:2503.24193*, 2025.

[18] B. G. Patra, D. Das, and S. Bandyopadhyay. Retrieving similar lyrics for music recommendation system. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, 2017.

[19] S. Rajput, N. Mehta, A. Singh, R. Hulikal Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Tran, J. Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[21] S. E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *TREC*, 1999.

[22] M. Schedl, S. Brandl, O. Lesota, E. Parada-Cabaleiro, D. Penz, and N. Rekabsaz. Lfm-2b: A dataset of enriched music listening events for recommender systems research and fairness analysis. In *Proceedings of the 2022 Conference on Human Information Interaction and Retrieval*, 2022.

[23] X. Su. A survey of collaborative filtering techniques, 2009.

[24] R. Surana, A. Namburi, G. Mundada, A. Lal, Z. Novack, J. McAuley, and J. Wu. Musicrs: Benchmarking audio-centric conversational recommendation. *arXiv preprint arXiv:2509.19469*, 2025.

[25] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[26] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

[27] O. Weller, M. Boratko, I. Naim, and J. Lee. On the theoretical limitations of embedding-based retrieval. *arXiv preprint arXiv:2508.21038*, 2025.

[28] S. Wu, Z. Guo, R. Yuan, J. Jiang, S. Doh, G. Xia, J. Nam, X. Li, F. Yu, and M. Sun. Clamp 3: Universal music information retrieval across unaligned modalities and unseen languages. *arXiv preprint arXiv:2502.10362*, 2025.

[29] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[30] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A In-Context Information for Inference

In the proposed framework, the recommendation is made based on the in-context information provided to the LLM. This conditioning prompt includes four types of information

System Prompt: The two stages of our framework have two distinct prompts: one for tool calling and, the other for response generation. For tool calling, we design a structured three-stage prompt for tool calling that decomposes the complex recommendation process into planning, retrieval, and reranking phases. Stage 1 (Planning) requires the LLM to select the exact retrieval tool and the reranking tool with a rationale for each choice. Stage 2 (Retrieval) executes the selected retrieval tool to gather at least `topk` unique `track_ids` from the music database. Stage 3 (Reranking) applies the selected reranking tool to reorder the candidates to improve recommendation quality. The workflow enforces strict constraints: tools must be used in sequence (retrieval $\rightarrow$ reranking) with complementary roles and no functional overlap. The response generation prompt is designed to clearly explain the recommendation results, ensuring that the recommended track aligns with user queries.

Tool Functions: We provide a comprehensive list of tool functions with their JSON schema that includes names, descriptions, and parameter data types. Detailed function examples are included in the Appendix B.

User Profiles: For personalization, we provide user demographics and recent listening history as in-context information to the LLM. The demographics include User ID, type, age group, and gender. The User ID serves as in-context information for inference to properly activate the "User-to-Item" tool. For recent listening history, we include metadata from the five most recent tracks from the user's listening session, where each track's information includes metadata, attributes, and Semantic IDs.

Previous Conversation History: We provide the previous conversation state as in-context information to the LLM. The conversation state includes the user queries, the recommended musics, and the responses from the LLM. To incorporate multimodal information of music, we include not only track IDs but also metadata, attributes, and Semantic IDs as in-context information.

B Tool Calling Functions

This appendix specifies the callable tools used by the agent in a Pythonic format. Each block presents the function signature, purpose, input arguments, and expected return. All tools return a list of `track_id` strings (up to `topk`), and are designed to be composed in sequence within the retrieval–reranking pipeline.

```python
def sql(sql_query: str, topk: int) -> list[str]:
    """
    Execute an SQL query for boolean (structured) matching.
    Must return results with track_id
    (e.g., `SELECT track_id FROM tracks WHERE ...`).
    SQL schema (table: tracks):
        track_id TEXT PRIMARY KEY
        title TEXT
        artist TEXT
        album TEXT
        popularity INTEGER
        release_date DATE (YYYY-MM-DD)
        tempo REAL
        key TEXT

    Args:
        sql_query: SQL query string to execute.
        topk: Maximum number of track_ids to return.

    Returns:
        List[str]: Up to `topk` track_ids.
    """
```

```python
def bm25(query: str, corpus_type: str, topk: int) -> list[str]:
```

```
    """
    Perform BM25 retrieval for lexical matching.
    Lowercase all input strings.

    BM25 corpora:
        'title'      : lowercase of track name
        'artist'     : lowercase of artist name
        'album'      : lowercase of album name
        'lyrics'     : lowercase of lyrics
        'attributes' : lowercase of genre, instrument, mood, theme, usage, etc.

    Args:
        query: Search query string.
        corpus_type:
        One of {"title", "artist", "album", "lyrics", "attributes"}.
        topk: Maximum number of track_ids to return.

    Returns:
        List[str]: Up to `topk` track_ids.
    """
```

```
def text_to_item_similarity(query: str, modality_type: str,
                            vector_db_type: str, topk: int) -> list[str]:
    """
    Perform text-to-item semantic similarity retrieval.

    Args:
        query: Search query string.
        modality_type: One of {"text", "audio", "image"}.
        vector_db_type: One of {"metadata", "lyrics",
        "attributes", "audio", "image"}.
        topk: Maximum number of track_ids to return.

    Returns:
        List[str]: Up to `topk` track_ids (most similar to the text query).
    """
```

```
def item_to_item_similarity(track_id: str, modality_type: str,
                            vector_db_type: str, topk: int) -> list[str]:
    """
    Perform item-to-item similarity retrieval (example-based recommendation).
    Note: `track_id` is a 22-character string.

    Args:
        track_id: Unique track identifier.
        modality_type: One of {"audio", "image", "cf"}.
        vector_db_type: One of {"audio", "image", "cf"}.
        topk: Maximum number of track_ids to return.

    Returns:
        List[str]: Up to `topk` track_ids similar to the input item.
    """
```

```
def user_to_item_similarity(user_id: str, topk: int) -> list[str]:
    """
    Perform user-to-item similarity retrieval (personalization).
    Use only the `user_id` from demographic/profile info.
    If `user_type` is "cold_start", do not select this tool.

    Args:
        user_id: Unique user identifier (string).
        topk: Maximum number of results to return.

    Returns:
```

```
        List[str]: Up to `topk` personalized track_ids.
    """
```

```
def semantic_id_matching(modality_type: str, indices: list[int],
                         topk: int) -> list[str]:
    """
    Perform Semantic ID matching via codebook index lookups.

    Args:
        modality_type: One of {"audio", "image", "metadata", "lyrics",
        ↪  "attributes", "cf_item"}.
        indices: List of code indices (e.g., Residual VQ codes).
        topk: Maximum number of results to return.

    Returns:
        List[str]: Up to `topk` track_ids matched by Semantic IDs.
    """
```

C Tool Environment

Our environment consists of an executor and databases for tool calling. For SQL and BM25 [21], we use basic metadata (track name, album name, artist name) and last.fm genre/style annotations from LFM-2b [22]. For dense retrieval tools, we construct multiple vector databases using various pretrained models: Qwen3-0.6 embedding [30] for text modality, CLAP [29] for audio modality, SigLIP2 [25] for image modality, and Bayesian Personalized Ranking (BPR) [20] for user and item embeddings. Finally, for Semantic IDs [19, 14], we train a separate Residual Vector Quantizer (RVQ) on pre-extracted each representations (listening history, metadata, semantic tags, lyrics, album art, and audio), and then combine them to generate multimodal semantic identifiers. The RVQ architecture employs 4 residual quantization layers with 64 codebooks per layer, optimized with commitment loss. The hyperparameters were selected through validation experiments designed to balance the commitment loss and codebook utilization. To ensure fair evaluation and prevent data leakage, we employ chronological data splits for training both the BPR and RVQ models, where trainset strictly precedes testset.

D Results

Multi-turn Recommendation: Table 2 demonstrates that our tool-calling approach outperforms baseline methods in zero-shot conversational music recommendation. The improvement in Hit@1 performance (0.022 vs 0.018 for Qwen3-LM + BM25) highlights the effectiveness of reranking through multiple tool integration. When a tool call once fails during inference (as discussed later in detail), our system automatically retries the call, ensuring that all inference steps are completed successfully.

Tool Calling Frequency: Figures 2 and 3 analyze tool calling patterns based on the first attempt before any retry mechanisms. After repeated retries, in our experiment, the system always reaches to extract a result. The tool frequency distribution reveals that natural language-friendly tools like SQL and BM25 show higher usage frequencies, while item-specific modalities such as item-to-item matching and semantic ID tools exhibit lower frequencies. This pattern likely reflects pretraining exposure: universal operators such as SQL and BM25 commonly appear across information retrieval corpora, whereas item-to-item matching and Semantic ID tools tend to have domain-specific function names and interfaces. Consequently, pretrained LLMs are less familiar with these tools and invoke them less frequently.

Tool Calling Success Rate: However, the success rate analysis reveals performance challenges with more complex tools. SQL queries achieve only a 27.4% success rate due to their syntactic complexity and common errors such as incorrect SQL syntax, using invalid column names, and retrieval failures caused by synonyms or typos leading to metadata-query mismatches. Item-to-item matching shows a 68.4% success rate, which can be attributed to the challenge of predicting a complete track ID, which represents private information that is related to the music catalog. Unexpectedly, tools that rely on

Table 3: An inference example with the LLM ***inputs*** and ***outputs***.

Input: User Demograph

UserID: N/A (Cold User), Age group: 20s, Country: Slovenia, Gender: Female, Previous Tracks: N/A (Cold User)

Input: Chat Context (Previous Turns)

User: I'm looking for a German electronic artist, I think they had an album called 'Walls'
with some really minimal and melancholic tracks.

Assistant: TrackID: 1KsqDnRQXFdFypQdFkB0wA,
title: not a number, artist: apparat, album: walls, tags: experimental, 00s, post-rock, instrumental, soundtrack
ambient, electronica, electronic, tempo: 130.37, key: C major, release_date: 2007.05.25
'audio:semanticID': [0, 39, 63, 53], 'image:semanticID': [41, 10, 17, 48], 'metadata:semanticID': [15, 49, 15, 36],
'lyrics:semanticID': [45, 38, 39, 50], 'attributes:semanticID': [20, 55, 58, 51], 'cf_item:semanticID': [59, 21, 42, 34]}

Assistant: Oh, absolutely! Apparat from the Walls album is exactly what I was thinking.
For a really minimal and melancholic vibe, you should definitely check out Not A Number.
It perfectly captures that atmospheric, introspective feeling.

User: Yes, this is exactly the artist I was looking for! Perfect, Apparat is the one.
Can you recommend some other tracks from their discography, especially more instrumental or ambient ones?

Intermediate: Chain-of-thought for Tool Calling

Okay, the user just confirmed that Apparat is the artist they were looking for and wants more recommendations
from their discography, specifically instrumental or ambient tracks. Let me break down how to approach this.
First, the user is a cold_start user, so I need to use the appropriate tools.
...[Omitted]...
Alternatively, using the 'artist' corpus_type with BM25 to find tracks by Apparat, then filter for instrumental,ambient.
But the tools available include BM25, which is for lexical matching. Alternatively, using text_to_item with
modality_type 'text' and corpus_type 'attributes', querying for instrumental, ambient.
...[Omitted]...
So the first tool call would be BM25 with query 'Apparat', corpus_type 'artist', topk 20.
Then, the second tool call would be text_to_item_similarity with query 'instrumental, ambient',
modality_type 'text', corpus_type 'attributes', topk 20. But the user is looking for tracks from Apparat,
so the initial retrieval is tracks by Apparat, and the reranking is based on attributes.

Intermediate: Tool Calling Results

[{"tool_name": "bm25","tool_args": {"query": "Apparat",corpus_type": "artist","topk": 20},
{"tool_name": "text_to_item","tool_args": {"query": "instrumental, ambient","modality_type": "text",
"corpus_type": "attributes","topk": 20}]

Output: Recommendation Track

Assistant: TrackID: 40HHuqH12ssXNpnHkiz0fk,
title: tod, artist: apparat, album: krieg und frieden (music for theatre)
tags: experimental, ambient, dance, electronic, soundtrack, tempo: 85.83, key: A major, release_date: 2013.02.19

Intermediate: Chain-of-thoguht for Response

Okay, the user just confirmed that Apparat is the artist they were looking for and asked for more recommendations,
specifically instrumental or ambient tracks. The track provided is Tod from Krieg und Frieden (Music for Theatre).
I need to verify if Tod is indeed instrumental or ambient. The tags include experimental, electronic, and soundtrack.
The tempo is 85.83, which is slow, and the key is A major. The release date is 2013.
...[Omitted]...
I should present this track as a recommendation, highlighting the genre, mood, and why it's a good match.
I need to make sure the response is enthusiastic and confident, confirming that this track matches their preferences.
Then, invite them to explore more similar tracks or discuss other preferences.

Output: Response

Assistant: Absolutely! Tod from Krieg und Frieden (Music for Theatre) is a stunning ambient track that perfectly
complements the minimal, introspective vibe you love. It's a slow, atmospheric piece with a haunting, melancholic
that feels both electronic and deeply emotional. The A major key and steady tempo create a meditative,
immersive experience. Would you like more recommendations or explore other moods?

unseen information types, such as User IDs and Semantic IDs that were never encountered during LLM pretraining, demonstrate high success rates. User-to-Item personalization achieves a 98.8% success rate, and Semantic ID retrieval shows a 95.8% success rate. This high performance stems from the rich in-context information provided through user profiles and previous conversation history.

Qualitative Example Table 3 shows an inference example of TalkPlay-Tools. The inference process begins with user information and conversational chat context, including previously recommended tracks and user profile data. The system proceeds through a tool calling stage where the model analyzes the user query and identifies appropriate retrieval tools. The retrieved results are then used to generate natural language response. This multi-stage process ensures both retrieval accuracy and conversational fluency in the final response.

E Conclusion

We presented a novel tool-calling framework for conversational music recommendation that seamlessly integrates multiple modalities and the various components in a retrieval-reranking pipeline. For successful integration, we proposed orchestrating external executors and models through LLM tool calling. Using various types of tools, the proposed framework effectively harnessed multimodal information spanning text, audio, visual, and behavioral data to deliver enhanced music recommendations within conversational contexts. Future research directions present several promising avenues: (1) incorporating reinforcement learning methodologies to optimize tool calling precision and minimize reliance on retry mechanisms; (2) designing personalized tool calling strategies that extend beyond track-centric approaches to capture nuanced user preferences and behavioral patterns more effectively.