Adding the expert touch: Formulating Expert-Driven Reward Functions for RL-Based Playlist Generation

Swetha Balaram

Delft University of Technology

TU Delft

XITE
music videos

Adding the expert touch: Formulating Expert-Driven Reward Functions for RL-Based Playlist Generation

by

Swetha Balaram

to obtain the degree of Master of Science in Computer Science
at the Delft University of Technology,
to be defended on Wednesday July 30, 2025.

Student Number:	6019552
Exam Committee:	Asst. Prof. Dr. Luciano Cavalcante Siebert, TU Delft
	Asst. Prof. Dr. Masoud Mansoury, (Asst. Prof., MMC), TU Delft
	Antonio Mone, (PhD Cand., IIG), TU Delft
	Dr. Zoltán Szlávik, XITE
	Dr. Ralvi Isufaj, XITE
Project Duration:	October, 2024 - July, 2025
Faculty:	Faculty of EEMCS, Delft

Preface

This thesis marks the completion of my Master's journey at TU Delft, one that has been as challenging as it has been rewarding. While my academic journey has been one of continuous learning and growth, it would not have been possible without the guidance, support, and encouragement of many wonderful people.

First and foremost, I would like to express my sincerest gratitude to my supervisor from TU Delft, Dr. Luciano Cavalcante Siebert. His constant encouragement and insightful feedback were instrumental in navigating the complexities of this research. His support gave me the confidence to explore new ideas, while also offering guidance to refocus my research when necessary. I am also grateful to Antonio Mone from TU Delft, whose timely insights and clear direction helped me whenever I was at a crossroads with the thesis.

I am incredibly grateful to my supervisors at XITE, Dr. Zoltán Szlávik and Dr. Ralvi Isufaj, for grounding this research in a practical, industry-focused context. Their guidance was a constant source of encouragement, and I will fondly remember our meetings, which often ended with a bunch of ideas drawn on a whiteboard. My heartfelt thanks also go to the music experts at XITE who participated in the interviews; your willingness to share your invaluable knowledge was the cornerstone of this project. I would also like to extend my thanks to Dr. Masoud Mansoury from TU Delft for serving on my thesis committee and for his time in reviewing my work.

This journey was made all the more memorable and fun by the friends I made along the way. To my friends here in Delft and back in India, thank you for the laughter, support, and encouragement that helped me through the pressures of academic life and in moments of doubt. Finally, my deepest and most heartfelt appreciation goes to my family. Your unconditional love, patience, and belief in me have been my greatest source of strength and confidence. Thank you for everything.

Swetha Balaram
Delft, July 2025

Abstract

Automatic theme-based playlist generation systems often fail to replicate the quality of expert human curation. While Reinforcement Learning (RL) offers a framework for this sequential task, its effectiveness is limited by the challenge of designing reward functions that capture the knowledge of professional curators. This thesis introduces and evaluates a methodology to bridge this gap by using Large Language Models (LLMs) to translate curatorial principles, gathered from expert interviews, into dense reward function code. The main aim of this research is to determine if LLMs can effectively interpret the complex strategies of professional curators and, in turn, guide an RL agent to produce playlists that adhere to expert standards.

To investigate this, we interviewed music experts and then used LLMs to create reward functions in two ways: one from a concise summary of the interviews and another from the complete raw transcripts. These reward functions were used to train a RL agent for playlist generation. The agents' performances were then evaluated for recommendation accuracy and alignment with the expert's curatorial style, and compared against two baselines: a similarity-based model and an RL agent with a hand-crafted reward function

The results showed that, the impact of the addition of the interview summarization step on the models' recommendation accuracy depended on the LLM, with the GPT-based model showing a significant increase in accuracy, while the Gemini-based model's performance remained consistent across both inputs. Furthermore, qualitative analysis of the generated reward functions revealed that the summarized transcripts resulted in high-level reward factors consistent across all the LLMs, whereas raw transcripts resulted in more varied and granular reward factors. Additionally, the choice of LLM impacted the final reward structure and the agent's subsequent performance. When compared against the baseline models in the cold-start scenario, RL agents guided by LLM-generated rewards significantly outperformed both the manually-tuned RL baseline and the non-RL similarity-based model. However, in seeded playlist continuation tasks, this performance hierarchy changed, with the simpler similarity-based model achieving higher recommendation accuracy.

Contents

List of Figures

List of Tables

1 Introduction

Music playlists, which are a list of songs organized around a theme, mood, or style, have reshaped how listeners discover and enjoy music in the digital era. Studies have shown that playlists account for a substantial portion of music streaming time in various streaming platforms, thereby increasing music consumption and growth of these platforms[1, 2]. Streaming services, recognizing this shift, have positioned playlists as a primary mode of music discovery and consumption. Central to the appeal of these streaming platforms is personalization, with users increasingly seeking experiences tailored to their tastes [3, 4].

Historically, professional music editors, often working at record labels, radio stations, or streaming services, have hand−picked tracks for flagship playlists. These experts are good at weaving tracks together to evoke specific moods, tell stories, or introduce listeners to new, fitting music within a particular theme [5]. This “expert touch“, often driven by years of experience, cultural understanding, and deep musical knowledge, is highly valued. However, manual expert curation is time-consuming and difficult to scale to meet the ever-growing demand for diverse, themed playlists [6, 7]. By contrast, largescale streaming algorithms typically rely on quantitative signals such as past play counts, skip rates, or metadata similarity. These systems optimize for short-term engagement: keep the user listening, minimize skips, and maximize “time spent“. However, purely algorithmic approaches to playlist generation often fall short in replicating the nuance and quality of human expert curation [8]. For instance, flagship editorial playlists on major platforms, e.g. Rap Caviar on Spotify, often retain immense popularity, underscoring the continued importance of human insight alongside algorithmic recommendations [9]. These systems can struggle to create truly engaging playlists, often reinforcing existing preferences, leading to “filter bubbles“ [10] and limiting musical diversity and homogenizing listening experiences over time [11, 12].

The challenge lies in capturing the tacit knowledge that guides human experts. Professional curators rely on years of musical experience, cultural context, and instinctive judgments. Much of this expertise is *tacit*—that is, it lives in their heads as intuitive, unspoken rules rather than explicit checklists. These explanations are rich in natural language but difficult to translate directly into numbers or code [13]. In fields like knowledge engineering, this challenge is known as the “knowledge acquisition bottleneck” [14, 15].

Reinforcement Learning (RL) [16] offers a promising framework for this problem. RL is a branch of machine learning in which an “agent“ learns to make sequential decisions through trial and error, guided by a numerical *reward function* [17]. Over time, the agent explores different actions, receives feedback (positive or negative rewards), and adjusts its strategy to maximize the cumulative reward. Unlike static recommendation methods which focus on user−specific metrics (e.g., likes, skips, play) [18] and song metadata [19] for immediate rewards, RL can optimize for long-term objectives, such as recommending items that might give smaller immediate rewards but could result in better cumulative rewards over time [20].

Traditional approaches to reward design in RL are often manual, relying on hand-crafted rules or predefined quantitative metrics. However, RL systems often face the reward design problem, where designing a good reward for a creative domain can be notoriously difficult [21]. Such methods may fail to capture the subtleties of human preferences or expert logic, potentially leading to misaligned agent behavior [22]. A system rewarded solely for maximizing listens or minimizing skips might learn to create bland, predictable playlists filled with popular hits, failing to capture the novelty and narrative that human curators provide. The quantitative objective (e.g., maximize play count) does not fully capture the qualitative goal (e.g., create a thematically resonant playlist).

The difficulty of translating qualitative goals into quantitative rewards has led to the exploration of Large Language Models (LLMs) as a potential solution. Recent advances have demonstrated LLMs' ability to understand, process, and structure information from free-form human language [23]. Their capacity to translate natural language inputs into formal representations like code or logical rules has been successfully applied in various domains [24, 25]. While more recent research has explored using LLMs to generate reward functions in fields like robotics and gaming [26, 27], this approach remains unexplored for music playlist generation and other creative domains like narrative generation, where expert tacit knowledge plays a significant role.

1.1. Research Questions

The challenge of integrating music playlist curators' tacit knowledge into automated systems motivates this thesis. We aim to explore how the tacit knowledge of music curators can be captured within a RL-based framework for playlist generation. This thesis is therefore driven by the following central research question:

RQ: How can we integrate expert playlist curators' knowledge into a RL system to generate theme-based playlists that reflects experts' curation standards?.

To provide a comprehensive answer, we further split our research question into two primary subquestions:

- **RQ1: How can we leverage LLMs to interpret professional curators' natural language explanations and generate executable dense reward function code for playlist creation?** LLMs been effective in generating code from natural language. This question investigates LLMs ability to convert music experts' tacit knowledge into dense reward function code.
- **RQ2: To what extent can an RL-based system that, guided by these expert-informed reward functions, generalize to create theme-based playlists that reflect expert curation principles?** Building on the first question, this question evaluates the performance of the RL agent trained on LLM-generated reward functions. It assesses whether an RL agent, guided by these rewards, can learn a policy that generates playlists reflecting professional standards. The performance of this agent is benchmarked against existing baseline models on the playlist generation task.

To address these questions, this thesis puts forth and evaluates a proposed solution: a system that integrates human expertise, LLMs, and RL. This system is developed and contextualized within **XITE**, an music video platform that relies on a team of experts for music playlist curation. Our proposed methodology involves eliciting playlist curation strategies through interviews with these experts, which are then translated by an LLM into a dense reward function. This expert-derived reward guides an RL agent to learn sequential playlist generation. We posit that LLMs can effectively formalize qualitative expert knowledge from interviews into reward function, allowing RL agents to learn expert-aligned playlist generations strategies.

Through our research, we make the following main contributions:

- We introduce a methodology for translating tacit expert knowledge into computable reward functions for RL agents. This approach uses semi-structured interviews to elicit qualitative principles from domain experts and leverages LLMs to automatically generate structured, multi-component reward code.

- We conduct a comparative analysis of two different knowledge processing pipelines for reward generation: one based on high-level, verified summaries of expert interviews and another using the anonymized, raw interview transcripts. Through our findings, we reveal a trade-off between the consistency of rewards derived from summaries and the granular, creative heuristics captured from raw text.
- We propose and implement a comprehensive, two-part evaluation framework for playlist generation systems. This framework assesses not only the alignment with ground-truth data using standard metrics (e.g., NDCG, Precision) but also evaluates the behavioral alignment of the agent by comparing the characteristics of generated playlists (e.g., Flow, Diversity, Novelty) against a quantitative ”expert profile”.
- We perform an empirical investigation into the application of different foundational LLMs (GPT-4, Gemini Pro, Claude) for the specific task of reward function generation for RL agents. The results provide insights into how model choice influences the structure and focus of the resulting reward functions.

1.2. Thesis Structure

This thesis is organized into seven chapters. Chapter 2 introduces the foundational concepts that would be used in the thesis such as RL, reward design problem, LLMs and existing evaluation methods. Chapter 3 goes into the existing research in music recommender systems, RL for recommendation, LLMs for knowledge extraction, and methods of eliciting tacit expert knowledge. Chapter 4 describes our proposed methodology, firstly exploring expert knowledge elicitation and the LLM-based pipeline that transforms interview transcripts into formal reward components. We then detail the design of the RL framework, including state/action representations, and the agent training process. Chapter 5 outlines our experimental setup: datasets, models being compared, implementation details, and the metrics used for the evaluations. Chapter 6 presents empirical results from the experiments demonstrating how well our system captures expert curation principles and compares to baselines. Chapter 7 details the broader implications of our results as well as the limitations of our work. Finally, Chapter 8 concludes by summarizing key findings and presenting directions for future research.

2 Background

2.1. Playlist Generation and Music Recommender Systems(MRS)

Music playlists are curated sequences of songs (tracks) designed to fulfill specific target characteristics, such as evoking a mood, adhering to a genre, or fitting a particular context [28]. In the current streaming landscape, they represent a primary mode of music discovery and consumption, accounting for over 30% of listening time [1, 29]. As given in [28], playlist generation is defined as:

"Given (1) a catalog of songs, (2) background knowledge, and (3) some target characteristics of the playlist, construct a sequence of songs fulfilling the target characteristics in the best possible way.".

The target characteristics usually indicate a theme that the songs in the playlist should adhere to. For instance, it could be based on a mood ('Happy Hits') or based on an artist ('Coldplay') or genres('Jazz').

Music recommender systems have evolved significantly, moving from traditional collaborative and content-based filtering to more sophisticated deep learning and graph-based approaches, increasingly incorporating multiple modalities of information to better understand user preferences and item characteristics [30]. A survey on music information retrieval (MIR) research on playlists done by Gabbolini et al. [31], categorized playlist generation two primary methodologies: Automatic Playlist Generation (APG) and Manual Playlist Generation (MPG) and further distinguished between playlists generated for individuals and for groups, based on the target audience. APG, which involves constructing a playlist via an algorithm, significantly reduces the time and effort of manual curation. A common task within APG is Automatic Playlist Continuation (APC), where an algorithm extends a playlist from a set of initial 'seed' tracks. The objective of these automatic methods, however, shifts depending on the target audience. When generating a playlist for a group of users (APG-G), for instance, the system must prioritize songs with a more general appeal that cater to the preferences of multiple members, rather than a single individual.

Various playlist generation methodologies have been explored, including similarity-based algorithms, collaborative filtering, content-based filtering and context-aware methods. Similarity is often derived from song content [32], metadata [33], tags, manually curated playlists, listening logs, and user ratings. Collaborative filtering (CF) techniques leverage historical user interactions to recommend songs based on preferences of similar users [34], while content-based(CB) approaches focus on analyzing audio features and metadata [19]. Context-aware methods consider additional factors such as user activity[35], mood[36], location of the user [37], or time of the day[38] to enhance playlist relevance. More recent advancements in MRS include deep learning techniques such as Autoencoders, RNNs[39], CNNs for audio feature extraction[40], graph-based methods like GNNs [41], node2vec [42, 43], and Session-Based Recommender Systems (SBRS) [44]. Despite these, many systems still generate playlists optimized for immediate engagement, potentially lacking in long-term satisfaction or narrative coherence.

2.1.1. The Role of Expert Curation

While algorithmic recommendations are scalable, the value of human expertise in playlist curation remains significant. Music experts, or curators, leverage deep domain knowledge, cultural context, and an intuitive understanding of musical aesthetics to create compelling listening experiences [5, 45]. This ”expert touch” is difficult to replicate with algorithms that primarily optimize for explicit engagement metrics like play counts or skip rates. Experts often start with a core set of songs and filter and arrange them to fit a theme, relying on their memory and experience where metadata is insufficient [46]. This qualitative, nuanced process has often resulted in highly popular and influential ”flagship” playlists on major streaming platforms, demonstrating that human insight is a critical component of perceived playlist quality [9, 47].

2.1.2. The Importance of Song Sequencing

The order in which songs appear in a playlist can impact the listener’s experience and the playlist’s perceived coherence. Effective sequencing is not just grouping similar tracks but involves creating a narrative or emotional arc [48, 28]. Smooth transitions between tracks, whether in tempo, energy, or mood, are known to significantly increase user satisfaction. Research into manually curated music albums reveals that artists often employ specific sequencing patterns, such as starting with higher-energy tracks and then alternating mood to prevent listener fatigue [49]. While that the exact ordering of songs does not drastically affect raw recommendation metrics, studies suggest that considering the context and properties of each track significantly improves next-song prediction and, by extension, user satisfaction [50]. Moreover, analyses of user-generated playlists confirm that adjacent tracks share stronger audio-feature similarity than random pairs, underlining the importance of sequence-awareness in playlist generation systems [51]. In short, a good playlist generation system must not only select the right songs but also arrange them in an engaging sequence.

2.1.3. Playlist Completion Task

The playlist completion task, often referred to as Automatic Playlist Continuation (APC), requires extending a partially-observed playlist by predicting the next k tracks that best fit the original theme or listening context. Formally, given a seed (initial) sequence of N tracks $(s_1, s_2, \ldots, s_N)$ drawn from a catalog $\mathcal{C}$, the goal is to produce an ordered list of k additional tracks $(s_{N+1}, \ldots, s_{N+k})$ such that the completed playlist maximizes song selection accuracy. This problem has been benchmarked in the RecSys 2018 Challenge on music playlist continuation [52] to evaluate the performance of different playlist generation systems.

The playlist completion task, often referred to as Automatic Playlist Continuation (APC), is a core problem in music recommendation. It challenges a system to extend a partially-observed playlist by predicting the next k tracks that best fit its original theme or listening context. Formally, given an initial ”seed” sequence of N tracks $(s_1, s_2, \ldots, s_N)$ from a song catalog $\mathcal{C}$, the objective is to generate an ordered list of k additional tracks $(s_{N+1}, \ldots, s_{N+k})$ that matches the songs present in the playlist considered as the ground-truth (reference-set). Due to its practical relevance and clear evaluation criteria, this task has become a standard benchmark for evaluating the performance of different playlist generation systems, notably in challenges like the RecSys 2018 Challenge [52].

Cold-Start in Playlists. A key difficulty in APC, and recommender systems in general, is the *cold-start* problem. This occurs when the system has insufficient data to make reliable inferences about user or item preferences [53, 54]. Without historical data, collaborative filtering approaches fail, forcing the system to rely on other signals. In playlist generation, this problem manifests in two primary ways:

- **User cold-start:** This refers to a situation where a new user interacts with the system. Lacking any listening history or explicit preferences for that user, the system cannot personalize recommendations effectively and may resort to suggesting globally popular or generic content.
- **Playlist cold-start (seedless):** This occurs when a playlist is generated from scratch, without any initial seed tracks to establish a context. The system must infer the intended theme solely from other available information, such as a playlist title or a brief description.

In both cold-start scenarios, the system’s ability to produce relevant recommendations is severely hampered. Overcoming this often requires leveraging content-based signals, such as audio features, lyrical content, or other metadata, to bridge the information gap [28].

Seedless vs. Seeded Playlist Generation. The APC task is typically evaluated in two distinct modes, each presenting a different set of challenges for the generation model:

1. **Seedless Generation:** In this mode, also known as $N = 0$ generation, the model must create a complete playlist of length k without any initial seed tracks. This evaluates the performance against the cold-start problem, as the model must rely entirely on abstract, high-level side information (e.g., a playlist title like "90s Pop" or a user profile) to infer the intended theme and generate a coherent sequence from nothing [28]. Success in this task demonstrates a system's ability to select and rank songs in accordance to a reference playlist.

2. **Seeded Generation:** In this mode, also known as $N > 0$ generation, the model is provided with the first N tracks of a playlist and is tasked with continuing the sequence by predicting the next k tracks. This shifts the problem from pure generation to contextual inference. The model must analyze the provided seed tracks to understand the established theme and then adapt its recommendations accordingly to create a seamless continuation [52]. The quality of the generated playlist in this scenario is highly dependent on the system's ability to accurately predict the next tracks based on the context provided by the seed.

2.2. Reinforcement Learning

RL is a sub-field of machine learning where an agent learns to make optimal decisions through direct interaction with an environment. The agent's goal is to maximize a cumulative numerical reward signal it receives over time [16]. This process of sequential decision-making is formally modeled as a Markov Decision Process (MDP). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the set of possible states the environment can be in. A state $s \in \mathcal{S}$ provides a complete description of the environment at a single point in time. The state space can be either discrete or continuous.
  - **Discrete State Space:** A finite or countably infinite set of distinct states. An example use case would be a robot navigating a maze, where the state would be the agent's specific location on the grid. For a finite number of states N, the space is written as:
  $$\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$$
  - **Continuous State Space:** An infinitely large set of states, typically defined as a subset of a multidimensional Euclidean space. An example application would be a self-driving car where the state is a high-dimensional vector composed of continuous variables like the car's exact speed, steering angle, and so on. Formally, it is written as:
  $$\mathcal{S} \subseteq \mathbb{R}^{D_S}$$
  where $D_S \in \mathbb{N}$ is the dimension of the state space.
- $\mathcal{A}$ is the set of all possible actions that an agent can execute. An action $a \in \mathcal{A}$ is a choice made by the agent. Like the state space, the action space can be discrete or continuous.
  - **Discrete Action Space:** A finite set of distinct actions. A typical use case would be making a robot navigate through a maze, with its possible actions would be to move 'up', 'down', 'left', or 'right'. For a finite number of actions M, this is written as:
  $$\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$$
  - **Continuous Action Space:** An infinite set of actions described by real-valued vectors. This is generally used in tasks like controlling the steering angle of a car. Formally, it is represented as a subset of a Euclidean space:
  $$\mathcal{A} \subseteq \mathbb{R}^{D_A}$$
  where $D_A \in \mathbb{N}$ is the dimension of the action space.

- $P(s' \mid s, a)$ is the transition probability function, indicating the probability of moving to state s' from state s after taking action a. This transition can be *deterministic*, meaning taking a specific action in a state always leads to the same next state or *non-deterministic* (or stochastic) environment which involves randomness in the state transition.
- $R(s, a)$ is the reward function, which returns a numerical reward after taking action a in state s.
- $\gamma \in [0, 1]$ is the discount factor, balancing the importance of immediate versus future rewards.

The agent's decision-making strategy is called a policy, denoted by *policy* $\pi(a \mid s)$, which maps each state s to a probability distribution over actions. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big].$$

Policies may be *on-policy*, where learning is based on actions taken by the current policy, or *off-policy*, where learning can leverage actions generated by a different (e.g. historical or exploratory) policy. RL has gained a lot of traction since studies have shown that an RL agent can perform extremely well in online gaming environments like chess [55], real-world simulations such as Gran Turismo Sport [56].

2.2.1. RL Algorithms and Proximal Policy Optimization

One of the more influential factors when training an RL agent can be the choice of training algorithm. Before explaining the algorithm used in our research, we briefly introduction into the different kinds of RL algorithms. An non-exhaustive taxonomy of algorithms is provided in Figure 2.1.

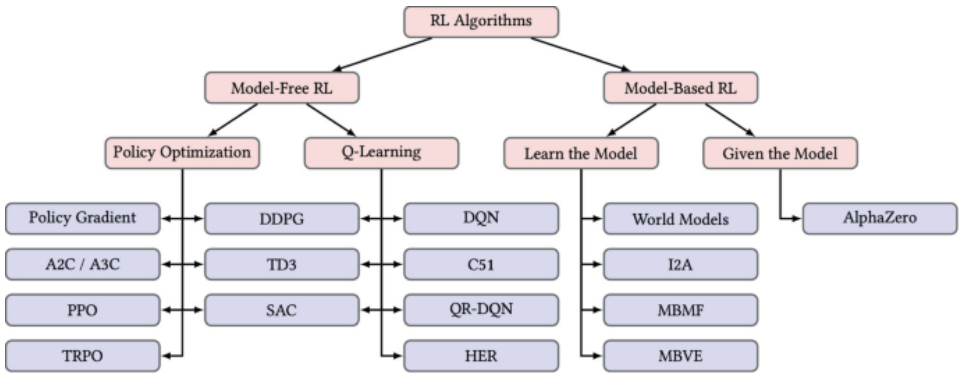


Figure 2.1: A non-exhaustive taxonomy of algorithms in RL. Img Source: [57]

While dividing the RL algorithms, one of the most prominent questions is whether there is some kind of function (or model of the environment) available that can help the agent in predicting the state transitions and rewards (model-based and model-free). In most cases, model-free algorithms are chosen over model-based, since access to ground-truth model (or learning the model) of the environment is usually difficult or can induce bias in the agent [58].

RL algorithms can be broadly categorized into value-based and policy-based [16]. Value-based RL algorithms like Q-Learning, State-Action-Reward-State-Action (SARSA), Deep Q-Networks (DQN), learn a value function that estimates the expected cumulative reward from each state or state-action pair. While these methods are often easier to implement, they tend to struggle in continuous or high-dimensional action spaces, since policy is determined by selecting actions that maximize the estimated value. Policy-based RL methods (Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO)), on the other hand, directly try to learn the policy by the policy's parameters (which map states to actions), to maximize expected return. In recommender systems, policy-based approaches [59, 60, 20] are generally preferred over value-based approaches in large state-action spaces, since they evaluate the policy instead of calculating the value over all actions. Proximal Policy Optimization (PPO) [61] is a policy gradient algorithm that has gained popularity for large-scale tasks, since it is less computationally demanding than other policy-gradient algorithms and can be applied to both discrete and continuous environments [62, 61]. The core of PPO is its policy objective, which aims to create a strategy that leads to higher rewards.

2.2.2. Large Action-Space Problem in RL

A significant challenge in applying RL to recommender systems is dealing with vast, discrete action spaces [59]. The set of possible actions (e.g., the entire catalog of songs) can easily number in thousands, resulting in the agent having to explore millions of state-action pairs. This makes exploration inefficient and policy learning computationally expensive. This computational expense arises from the lengthy hyperparameter search followed by several full training runs of the RL agent, each demanding typically millions of timesteps to ensure the agent's policy is both effective and statistically reliable. Additionally, previous studies like [63] have shown that it is difficult to design variable action spaces for different states. Several strategies exist to manage this:

- **Invalid Action Penalty:** Assigning a negative reward for invalid actions [64]. This is often inefficient as the agent must first learn to avoid these penalties through trial and error, which does not scale well.
- **Action Space Shaping:** Reducing the action space by removing non-useful actions or discretizing a continuous space [65, 63]. A downside to this approach is that it can be difficult to implement without losing potentially optimal actions [59].
- **Invalid Action Masking:** This is a more direct and effective approach where the probabilities of all invalid actions for a given state are masked (i.e., set to zero) before the action selection step [66]. This strictly prevents the agent from selecting invalid actions (like a song already in the playlist or a song that does not fit the theme). This technique has been successfully applied in complex domains with large discrete action spaces, such as real-time strategy games [67, 68], and is the most suitable method for our task as it directly prunes the decision space to only valid candidates at each step.

2.2.3. Reward Design Problem in RL

The reward function is the most critical component in an RL system, as it defines the task and guides the agent's behavior. The principle of ”reward is enough” posits that the maximization of a well-defined reward signal can lead to the emergence of all desired intelligent behaviors [22], while an inadequately defined reward signal can negatively impact the agent's performance [69]. However, designing such a function, a process known as reward engineering, is notoriously difficult [21].

Reward functions are typically classified into two types:

- Sparse Rewards: Feedback is provided infrequently, often only upon completion of the entire task (e.g., a final score for a complete playlist). This makes learning very difficult, as the agent struggles to attribute a final outcome to the long sequence of actions that led to it. This is known as the credit assignment problem. Techniques like Hindsight Experience Replay (HER) have been developed to help agents learn from unsuccessful attempts in sparse reward settings [70].
- Dense Rewards: Feedback is provided at each step, guiding the agent with more frequent signals. This generally accelerates learning but carries the risk of specifying a flawed reward that leads to unintended ”reward hacking,” where the agent optimizes the proxy metric without achieving the true goal [69]. For instance, a reward for song popularity might lead to playlists that are popular but lack novelty or coherence. Manually crafting dense rewards that capture nuanced human preferences has been a significant challenge [71].

2.3. Tacit Knowledge Elicitation and LLMs

Expertise, particularly in creative domains like music curation, is heavily reliant on tacit knowledge. This form of knowledge, as defined by philosopher Michael Polanyi, is knowledge that we “know more than we can tell“ [72, 73]. It is deeply personalized, context-dependent, and acquired through years of experience and practice, making it difficult for experts to fully articulate or codify into explicit rules [14, 15]. This challenge of converting tacit knowledge into an explicit, computable format is a well-known problem in knowledge engineering, often referred to as the “knowledge acquisition bottleneck“.

Several techniques exist to elicit tacit knowledge, with interviews being a cornerstone method [73]. These can range from unstructured conversations to highly structured questionnaires. For capturing the nuanced, subjective logic of experts, semi-structured interviews are particularly effective [74, 75]. This format uses a set of guiding questions but allows the interviewer the flexibility to probe deeper, ask follow-up questions, and explore other sub-topics within the same area of concern based on the expert’s responses [76]. This balance of structure and freedom works well for uncovering the underlying ”why” behind an expert’s decisions, making it a powerful tool for understanding the tacit dimensions of creative processes [77, 78].

2.3.1. Large Language Models(LLMs)

Large Language Models (LLMs) are a class of deep learning models, most commonly based on the transformer architecture [79], that are pre-trained on vast quantities of text and code [80, 81]. This extensive training enables them to acquire a sophisticated understanding of grammar, syntax, semantics, and real-world concepts. Models like OpenAI’s GPT-4 [82], Google’s Gemini [83], and Anthropic’s Claude [84] have demonstrated remarkable capabilities in understanding, generating, and reasoning about human language.

Their ability to process unstructured text makes them uniquely suited to bridging the gap between qualitative human knowledge and quantitative computational systems. LLMs can analyze interview transcripts, identify thematic patterns, and even translate natural language descriptions of a task into formal, structured representations like code or logical rules [85, 24, 86]. While they differ in their specific architectures, training data, and fine-tuning methods (e.g., Gemini’s native multimodality or Claude’s training with ”Constitutional AI”), their core strength lies in capturing the rich context and intent embedded in human language, thereby offering a promising solution to the knowledge acquisition bottleneck.

2.4. Knowledge Graphs and Node2Vec

A knowledge graph (KG) is a structured representation of information where entities (nodes) are connected by their relationships (edges). In the music domain, a KG can represent songs, artists, genres, and decades as nodes, with edges representing relationships like ”performed by,” ”belongs to genre,” or ”released in decade” [64]. This creates a rich, interconnected network of musical knowledge.

To leverage this structure for machine learning, we need to convert the nodes into low-dimensional numerical vectors, or embeddings. Node2vec is a powerful algorithm for this task [43]. It generates embeddings by simulating biased random walks on the graph. The algorithm uses two parameters, p (return parameter) and q (in-out parameter), to control the walks. A low p encourages the walk to explore new nodes, while a low q keeps the walk localized. By exploring diverse neighborhoods, node2vec captures both local community structure (homophily) and broader structural roles of nodes. The sequences of nodes generated by these walks are then fed into a Skip-gram model, similar to Word2Vec, to learn the final embeddings. In recommendation systems, these embeddings have proven highly effective for tasks like finding similar items or users, often outperforming traditional baselines [42, 87].

In the domain of recommendation systems, node2vec can be employed to generate embeddings for users and items based on their interaction graphs (e.g., user-purchase histories, user-rating data). These embeddings can then be used to suggest relevant items to users or to find similar users and has been shown to outperform collaborative filtering baselines and other graph embedding methods in terms of recommendation accuracy [42, 87].

2.5. Statistical Significance Testing

In empirical research, statistical significance testing is essential to determine if observed differences in model performance are genuine or simply due to random chance. The process involves formulating a null hypothesis (H0), which assumes no true difference exists, and then calculating a p-value, indicating the probability of observing the results if the null hypothesis were correct. If this p-value falls below a predefined significance level (α), typically 0.05, the null hypothesis is rejected, and the result is deemed statistically significant.

The choice of statistical test depends on the data's characteristics. When comparing two models on the same set of evaluation instances, the performance scores are paired. If the differences between these pairs cannot be assumed to follow a normal distribution, a parametric test like the paired t-test is inappropriate. In such cases, the non-parametric Wilcoxon signed-rank test is the appropriate method to assess these differences [88]. It tests the null hypothesis that the median difference between paired observations is zero, without making strong assumptions about the underlying distribution of the data.

Another challenge that emerges is the multiple testing problem, where conducting numerous pairwise comparisons across multiple models increases the likelihood of a "false positive". To mitigate this, a correction for multiple testing, such as the Benjamini-Hochberg (BH) procedure controls the False Discovery Rate (FDR) [89]. The FDR is the expected proportion of claims of significance that are actually false. By adjusting the raw p-values from the entire family of tests, the BH procedure ensures that our overall findings remain statistically reliable.

3 Literature Review

This chapter examines prior literature related to the application of RL for music playlist generation. The analysis is structured to establish the suitability of RL as a sequential decision-making framework for this task. We cover foundational and more recent models, while focusing on the critical approaches to reward engineering, state representation, and action space management. By examining the methodologies, strengths, and limitations of existing approaches, we focus on gaps that we aim to address through our work.

3.1. RL-based Playlist Generation Systems

In this section, we cover the existing works that have been done in RL in the context of playlist generation systems. Central to any RL application is the formal definition of its environment, which influences the information provided to the agents and as a result the actions it would pick in a given state. This section delves into how the state and action spaces, are represented in existing literature of playlist generation systems, reviewing various strategies that researchers have employed to model a playlist and the catalog of candidate songs.

3.1.1. State-Action Space Representation

Reinforcement learning (RL) offers an alternative to traditional methods by learning playlist generation models that directly optimize user satisfaction through interaction[90]. The efficacy of an RL agent depends significantly on how its environment, particularly the state and action spaces, is defined and represented. In context of playlist generation, previous works have explored a variety of representations. The state which encapsulates the current listening context, has been encoded as sequences of track identifiers or song embeddings [39, 90], song-feature summaries [91], or enriched embeddings from knowledge graphs [92].

The action space on the other hand consists of the candidate pool of songs to choose the next track from, typically represented by a discrete song id [39] or its feature/embedding vector [92]. However, this presents a challenge when the agent must choose from thousands of potential songs in a discrete action space. For example, a study by Dulac-Arnold et al. [59] addressed the challenge of large action spaces in RL by embedding actions in a continuous space using prior information. This approach allowed for generalization across similar actions and leveraged approximate nearest-neighbor (NN) methods to achieve logarithmic-time lookup complexity. This principle can be applied to playlist generation by representing songs in an embedding space where similar songs are located near one another.

Another notable method is the combination of hierarchical clustering with Q-learning proposed in [93], where clusters of songs serve as states rather than individual songs. This hierarchical approach allows for scalability in large libraries (100–1,000 songs) and has been shown to outperform baseline shuffle-mode systems. By focusing on audio characteristics instead of metadata, this method improves performance in offline settings without requiring explicit user input. However, this requires finding the similarity between songs and determining the right value for the number of clusters. Determining the similarity between songs can be subjective [31]. Simple metadata alone often fails to adequately determine whether a song fits the current state of a playlist in terms of genre, tempo, or mood[46]. Additional information such as the lyrics and audio characteristics could help understand the type of message the song is trying to convey. Song similarity can be measured using Mel-Frequency Cepstral Coefficients (MFCCs), learned embedding representations, metadata, or expert annotations. Knowledge graphs, where songs and metadata are represented as nodes and their relationships as edges, have also been used to model song similarity [94, 92]. These approaches enable a more nuanced understanding of how songs are related to one another.

3.1.2. Approaches Addressing Cold-Start Problem

As persistent challenge in playlist generation is the cold-start problem, previously introduced in 2. Oord et al. [40] proposed applying deep convolutional neural networks (CNNs) to the raw audio of songs to tackle the cold start problem for newly release songs which did not have enough user-interaction data or meta tags. By learning a representation directly from the audio signal, their model was able to recommend new songs that have no listening data, effectively solving the item cold-start problem. However, it does not solve the user cold-start problem, as it has no information about a new user's preferences. Additionally, since the model is trained to predict factors from a collaborative filtering system, it can cause popularity biases and not necessarily incorporate the theme of a playlist.

For the specific challenge of playlist continuation when the starting songs are few or unpopular, latent factor models offer a different solution. Yürekli et al. proposed using Latent Semantic Indexing (LSI) on a song-playlist matrix to find hidden relationships between tracks [54]. This technique improves recommendations in sparse data situations by inferring connections between songs that may not have appeared together before. While this alleviates the issue, it does not solve it entirely, as this approach treats a playlist completion as at “set-problem“ meaning it does not consider the sequencing of songs.

Addressing a more extreme case, seed-free generation, requires a different approach where a playlist is created from scratch. Li et al. proposed a generative model that learns the characteristics of well-formed playlists and can generate a new one based on high-level constraints like a requested genre or theme [95]. This directly tackles the problem of starting a playlist with no initial song. The main limitation is that, like the LSI model, this approach also tends to overlook the importance of sequence. Furthermore, its effectiveness relies heavily on the quality and completeness of the metadata tags used for conditioning, which could be an issue when there is lack of good metadata.

3.2. Graph-embedding (Node2Vec) in Music Recommendation

For playlist generation system, one of the more common issues lie when user-data is sparse, and when the metadata present in the song-database is limited, which the number of distinct values for each attributed not being enough to distinguish between songs. In such instances, knowledge graph can be a good way to capture information about the songs and relation between other song based entities like decade, genre, artists and so on. Graph-based embeddings have been used extensively to capture relationships between songs, artists, genres, and user preferences in music recommender systems [92, 41]. These embeddings allow for modeling rich semantic and structural connections, particularly in cases where the songs' metadata alone is insufficient for expressing user intent or song similarity.

One of the baselines for our work in representation of the songs and the RL agents state space is the Node2Vec model being used in [96]. Here, a collaborative knowledge graph (CKG) is constructed where songs, artists, genres, subgenres, decades, and curated playlists are all nodes, with edge weights reflecting the relative importance of each relation (e.g. song–genre, song–artist, song–playlist). To capture not only these metadata relationships but also acoustic similarity, they augment the CKG with direct song–song connections. For each track, they generate audio embeddings using pre-trained MULE (Musicset Unsupervised Large Embedding) [97], which was trained on log-mel spectrograms using a SimCLR contrastive objective over a massive Musicset corpus. For determining the connections to other songs, they used Faiss to find its ten nearest neighbors in a high-dimensional audio-embedding space, then added edges whose weights are set in proportion to the cosine similarity between the MULE embeddings. This allowed the model to traverse directly between sonically similar tracks, bypassing intermediate nodes like genre or artist. The edge weights itself was determined through scaling the edge weights, one can control exactly how much the audio features influence the graph structure.

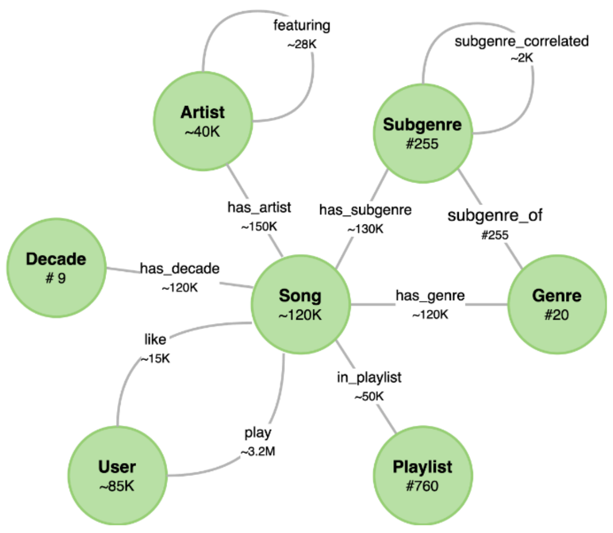


Figure 3.1: Graph schema for the full CKG with occurrence counts. Img Src: [96]

Node2Vec [43] is then applied to learn low-dimensional embeddings for every node in this enriched graph. For generating playlists using the Node2Vec model, the incomplete playlist at each step is represented by the sequence of its song-node embeddings. The next track is then chosen by finding the nearest neighbors of the last song's embedding and greedily extending the playlist. This simple neighbor chaining policy implicitly enforces both metadata and acoustic coherence. The obtained Node2Vec embeddings encode both song metadata and audio similarity, an approach we adopt as the backbone of our RL agent's state representation. On the playlist completion task, the Node2Vec model performs

3.3. Reward Function Formulation in Playlist Generation Systems

Traditional methods of reward function specification often rely on hand-crafted rules or predefined quantitative metrics, which does not effectively capture expert's metrics. Hu et al. aimed to enhance playlist generation by incorporating user feedback on recommendations into the reward function [98]. However, this method risks creating a feedback loop that could restrict users' exposure to diverse content.

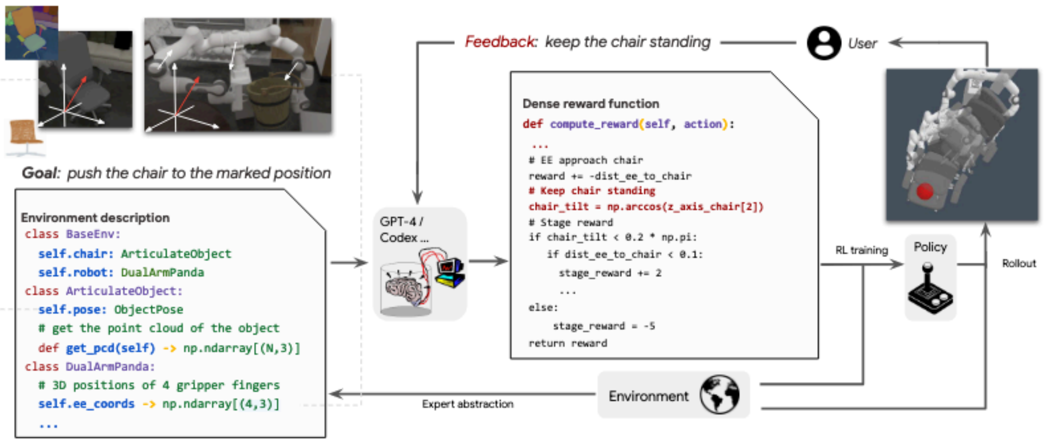
Authors Shih et al. model playlist generation as a language-modeling task refined by policy-gradient RL [39]. Their agent is an attention-RNN language model (song sequences as "words"), optimized with hand-crafted reward components. Specifically, they define four metrics (diversity, novelty, freshness, coherence) and combine them as a weighted sum: for example, diversity is the Euclidean distance between track embeddings, novelty is inversely related to playcount, freshness penalizes older release years, and coherence is the log-probability from the pretrained RNN. All reward terms are manually designed and weighted for each user preference. As a result, the system can flexibly tune between diversity or novelty, but it requires careful tuning of those metrics. In contrast, our approach avoids manual reward coding by using LLMs to interpret expert-specific preferences from interview text.

Another recent example is from industry: Spotify employed a simulation-based RL to create playlists for user satisfaction [90]. Here the state includes high-dimensional track features and user context; the action is choosing the next track, and rewards are derived from a user-simulator (e.g. positive reward if the simulated user plays a song, negative if skipped). They train a modified deep Q-network ("Action-Head DQN") in a model-based framework. This approach yields improved satisfaction metrics in offline A/B tests, but depends on the accuracy of the user model. Like Shih et al., it relies on predefined, explicit reward signals (user-play behavior). Furthermore, the reliance on simulated user behavior may introduce a gap in capturing an playlist curators' decision-making process. Other works have emphasized achieving smooth transitions between tracks by embedding acoustic similarity metrics into the reward function [99, 94]. Despite these varied approaches, there is still no clear agreement in the field regarding which reward function components yield the best results.

3.4. Natural Language to Reward Function

Recent studies have tried to address the issue of reward design in other RL-based tasks by introducing new approaches, specifically including Large Language Models(LLMs), that make reward design easier and flexible. One such study explores how LLMs can be leveraged as proxies for reward functions, converting textual descriptions of desired out- comes into quantitative signals [27]. This approach has the potential to significantly streamline reward design by making it more intuitive and aligned with curators' intention.

Building on [27], Xie et al. propose Text2Reward, which uses large language models to turn natural-language goals into dense reward functions for RL [26]. Given a textual task description (e.g. "push the chair to the marked position"), Text2Reward generates executable reward-code (in Python) that an RL agent can use.


Feedback: keep the chair standing
User
Goal: push the chair to the marked position
GPT-4 / Codex ...
RL training
Policy
Rollout
Environment
Expert abstraction

Environment description
```
class BaseEnv:
  self.chair: ArticulateObject
  self.robot: DualArmPanda
class ArticulateObject:
  self.pose: ObjectPose
  # get the point cloud of the object
  def get_pcd(self) -> np.ndarray[(N,3)]
class DualArmPanda:
  # 3D positions of 4 gripper fingers
  self.ee_coords -> np.ndarray[(4,3)]
  ...
```

Dense reward function
```
def compute_reward(self, action):
 ...
 # EE approach chair
 reward += -dist_ee_to_chair
 # Keep chair standing
 chair_tilt = np.arccos(z_axis_chair[2])
 # Stage reward
 if chair_tilt < 0.2 * np.pi:
    if dist_ee_to_chair < 0.1:
      stage_reward += 2
      ...
 else:
     stage_reward = -5
 return reward
```

Figure 3.2: An overview of TEXT2REWARD pipeline proposed in [26]

They demonstrate this on standard benchmarks (robotic manipulation in ManiSkill2/MetaWorld and locomotion in MuJoCo) and report that policies trained with the LLM-generated rewards perform on par with expert-designed rewards. This approach is fully automated (no manual tuning of reward weights) and operates on general control tasks with clear objective goals. We build upon their work in a more domain-specific setting: we derive reward functions from interviews about musical taste, which are more subjective and varied than the well-defined goals in Text2Reward. Thus, while Text2Reward validates the idea of text-to-reward shaping in RL, its tasks and assumptions differ substantially from playlist generation. Our method adapts the concept to a specialized music context, using interview text to inform the reward rather than generic task descriptions.

4 Methodology

This chapter details the methodology employed to develop an RL-based system for generating music playlists that align with expert curatorial standards. The core of our approach involves eliciting the nuanced, often tacit, decision-making principles of professional music playlist curators, translating this qualitative knowledge into a computable reward function to guide a RL agent for playlist generation. The methodology is divided into two main phases:

1. **Text to Reward:** Capturing curator decision-making through semi-structured interviews, processing the interview data and generating reward functions using LLMs via two input processing pipelines: one using summarized transcripts and the other using raw transcripts.
2. **Expert-Guided RL for Playlist Generation:** Defining the environment within which the RL agent learns, guided by the expert-informed reward.

4.1. From Text to Reward: Methodology for Eliciting Expert Knowledge

This section details the first part of our methodology, which includes converting qualitative expert knowledge into a reward function for training the RL agent. The approach begins with knowledge elicitation through semi-structured interviews. It then outlines two pipelines for processing the interview data: one using high-level summaries and the other using the raw transcripts. Finally, it describes how the processed text is converted into a reward function and code which is further refined through a feedback loop. Figure 4.1 givens an overivew of our text to reward process

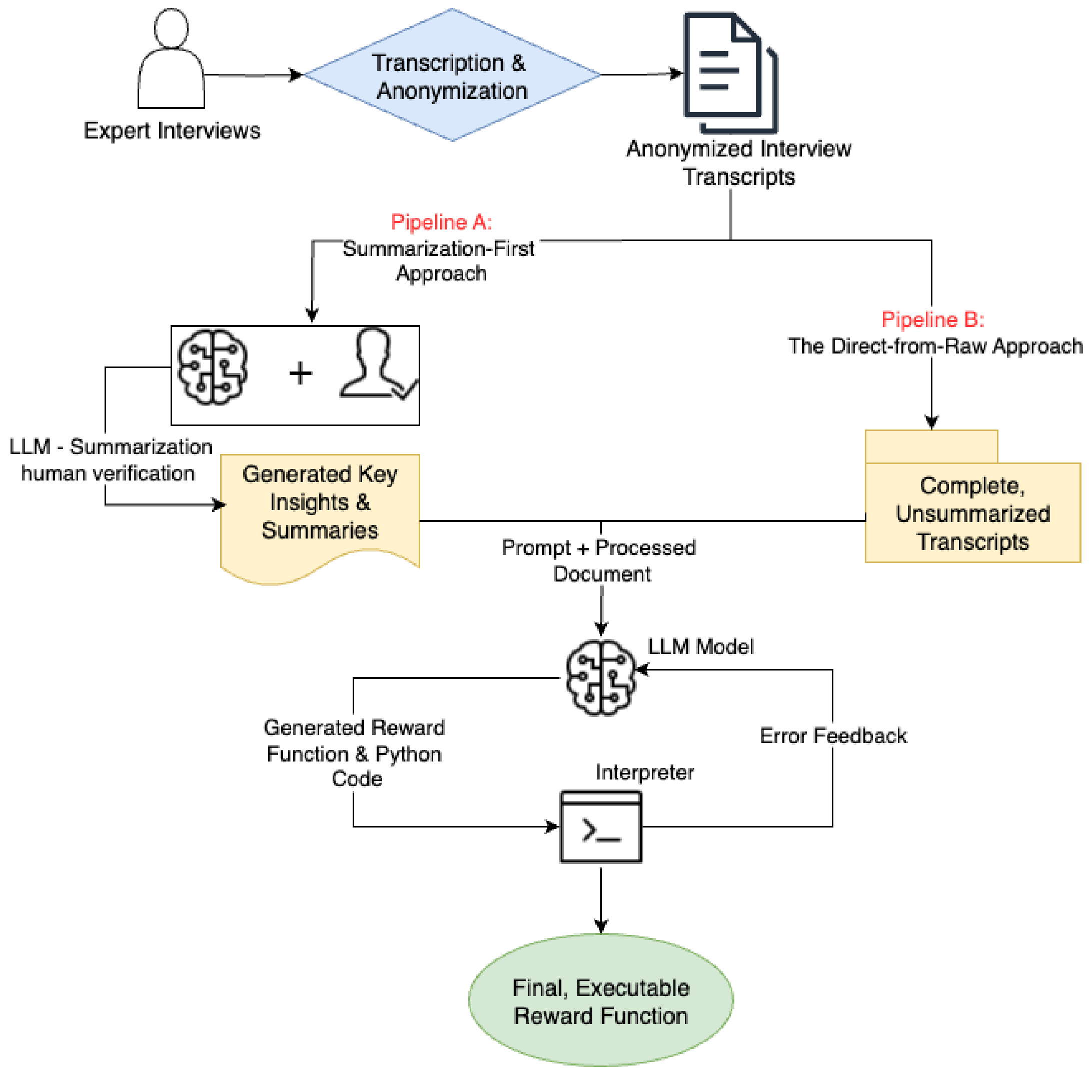

Expert Interviews
Transcription & Anonymization
Anonymized Interview Transcripts
Pipeline A: Summarization-First Approach
Pipeline B: The Direct-from-Raw Approach
LLM - Summarization human verification
Generated Key Insights & Summaries
Complete, Unsummarized Transcripts
Prompt + Processed Document
LLM Model
Generated Reward Function & Python Code
Error Feedback
Interpreter
Final, Executable Reward Function

Figure 4.1: An overview of the methodology for converting expert interview transcripts into an executable reward function.

4.1.1. Expert Knowledge Elicitation: Interview Design

For the interview process, we interviewed 8 experts from the XITE music team, who are responsible for curating various playlists on the platform. Each interview was conducted for an hour, resulting in 10-12 pages long transcripts per interview. Through the interviews, we tried to understand the experts' thought process, the factors considered, and the degree of importance given to each factor while selecting and ordering songs for a given playlist. To ensure that the experts can be as descriptive as possible and dive deep into their process while still guiding the discussion, we opted for a semi-structured format as introduced in Section 2 for the interviews. The interviews were designed to determine how experts balance factors like cohesion, diversity, smooth transitions, popularity, and so on. The interview protocol was organized around four core themes:

1. Initial song selection strategies for a new theme.
2. Principles of sequencing and song-to-song transitions.
3. The role of diversity, popularity, song metadata, acoustics and novelty within a coherent playlist.
4. The practical workflow while curating a specific playlist.

The questions targeted to uncover the criteria that the experts consider while selecting the songs that go into a specific playlist. For example, themes such as ”90's pop” or ”80's rock” were used as examples to elicit more general principles.

The interviews were recorded and transcribed using the company's internal tools: Google meet for the recording and transcription of the interviews. The names of the participants and other named entities (company names, artist names, client and countries) and any other PII (personally identifiable information) mentioned during the interview were anonymized before proceeding with the analysis and summarization.

4.1.2. Knowledge Processing: A Comparative Approach to Reward Generation

A central methodological question during this phase is determining the level of information granularity required to generate a high-fidelity reward function from qualitative data. Should the LLM be provided with a concise, high-level summary of expert principles, or should it reason over the full complexity and nuance of the raw transcripts?

To investigate this trade-off, we designed and compared two distinct pipelines for converting the textual data into a reward function.

Pipeline A: The Summarization-First Approach
This pipeline tests the hypothesis that a pre-processed, high-level summary allows the LLM to identify core, consensual principles more effectively, reducing noise from individual and elaborate transcripts.

1. LLM-Powered Summarization: Traditional qualitative analysis methods like LDA often fail to capture deep contextual meaning [100]. Inspired by recent work [75, 101], we first employed an LLM (Gemini Pro) to analyze and synthesize the anonymized transcripts from all eight experts into a single, cohesive summary. The Gemini Pro model has demonstrated strong performance in qualitative content analysis of interview data [102]. We used a zero-shot prompt instructing the model to extract key principles, rules, and heuristics, grouping them into categories supported by textual evidence.

2. Human-in-the-Loop Verification: Given LLMs' tendency to hallucinate their responses, [103, 104], the qualitative analysis and summary generated by the LLM was further verified manually, comparing it to the researchers' key notes and original interview transcripts taken during the interviews, to ensure that human insight was included in determining key information and correcting any information that was misinterpreted by the LLM. This human-in-the-loop step was critical for refining the output. Manual analysis of the provided summaries brought to light some discrepancies between the factors identified and the excerpts that served as base for the factors. For instance, *"Song Length Appropriateness"* was identified as one of the key factors, based on an excerpt from one of the interviews, *"A song of six minutes which is very very sad... maybe take that out because we could lose people"*. Additionally, there were repetitions in the factors identified. For instance, identified factors like "Song's adherence to primary genre/decade of the theme" and "Maintaining Genre Consistency" point toward the same criteria, which could lead to redundancies during the formulation of the reward function. Figure A.1 in the Appendix, shows a snippet of The qualitative analysis generated by the LLM is Adding human-insight to final summary ensured a more accurate representation of the experts' collective knowledge by correcting the errors and redundancies.

Pipeline B: The Direct-from-Raw Approach
This second pipeline operates on the hypothesis that providing the LLM with the complete, unsummarized transcripts allows it to capture more subtle, varied, and potentially contradictory heuristics that might be lost during abstraction. This approach trades the clarity of a summary for the variance of the original data. In this pipeline, the full set of anonymized raw transcripts was concatenated and used as the direct input for the reward generation step. The aim is to understand whether the LLM can process inputs from all experts and give a more generalizable reward function that acts as a mixture of experts.

4.1.3. From Processed Text to Reward Function

The final step in both pipelines is the conversion of the processed text (either from Pipeline A or from Pipeline B) into a computable reward function. Following a general approach inspired by Xie et al. [26], we engineered a detailed prompt that along with the inputs from Pipeline A or B, provided the LLM with three categories of information:

1. **Task and Persona Context:** The prompt began by assigning the model the role of an expert in RL and code generation and outlined the high-level goal of creating a RL-based agent to generate playlists aligned with human expert standards.
2. **Detailed Environment Specification:** To ground the code generation task and ensure compatibility with our existing framework, the prompt included the precise Python class definitions for the RL environment. This included the structure of the state and action spaces, available song metadata, and the mechanics of the action masking process which will be introduced in Section 4.2. Providing this concrete implementation detail was intended to minimize hallucination and syntactic errors in the generated code.
3. **Negative Constraints:** The prompt explicitly included negative constraints to guide the model away from redundant calculations. For example, specific instruction was provided to not include a reward component for theme alignment, as this is already enforced by the environment's action masker which only permits theme-relevant songs to be selected.
4. **Output Format:** To ensure the output is structured and interpretable, the prompt specifies a required format, instructing the model to provide the reward function as mathematical formula, followed by a detailed explanation of each component and its corresponding weight, and finally, to provide the logic within a predefined Python class structure.

4.1.4. Iterative Refinement and Code Execution:

Once the output is generated by the LLM, we implement an automated self-repair execution loop on the generated reward code, a technique validated in prior works on code generation [25, 105]. The LLM's initial code is executed in a sandboxed interpreter. Any resulting stderr traceback is captured and used as feedback to prompt the model for a correction. The feedback prompt contained the original code that failed, the captured error traceback, and an instruction for the LLM to debug the code and provide a corrected version. This iterative refinement process was repeated until the code was free of syntax and runtime errors. The initial code generated by the LLMs frequently contained a predictable set of errors, most commonly involving mismatches in song metadata column names, incorrect variable references, and data type conflicts. The self-repair process thus ensured that the final code on which the RL agent was trained was not only syntactically correct but also avoided runtime exceptions.

4.2. Reinforcement Learning for Playlist Generation

The playlist generation process is framed as a Markov Decision Process (MDP), where the state represents the current playlist and attributes of the included songs. This chapter details the design of this MDP, including the definitions of the state and action spaces, the formulation of the reward signal, and the specifics of the training algorithm and experimental setup. The overall training loop is depicted in Figure 4.2.

4.2.1. State and Action Space Representation

An important design choice is how to represent the current state of the playlist being built. An analysis into the song data showed that song metadata available was found to be insufficient, as it lacked the richness to distinguish nuanced acoustic differences between tracks. A snippet of the meta data provided in table 4.1 shows the limited categories for the song metadata types. Additionally, most tracks has NULL values for their metadata. To overcome this lack of information on the songs, we adopted a multi-component state representation centered on Node2Vec embeddings and popularity metrics, which was extended from the baseline model described in Section 3. These embeddings were derived from a collaborative knowledge graph, capturing complex relationships between songs, artists, genres, and user interactions.

Generating Song Embeddings via Weighted Node2Vec

To capture the various dimensions of similarity between songs and the metadata (thematic, acoustic, and collaborative), we built upon Node2vec approach introduced in Section 3. We adopted a two-stage

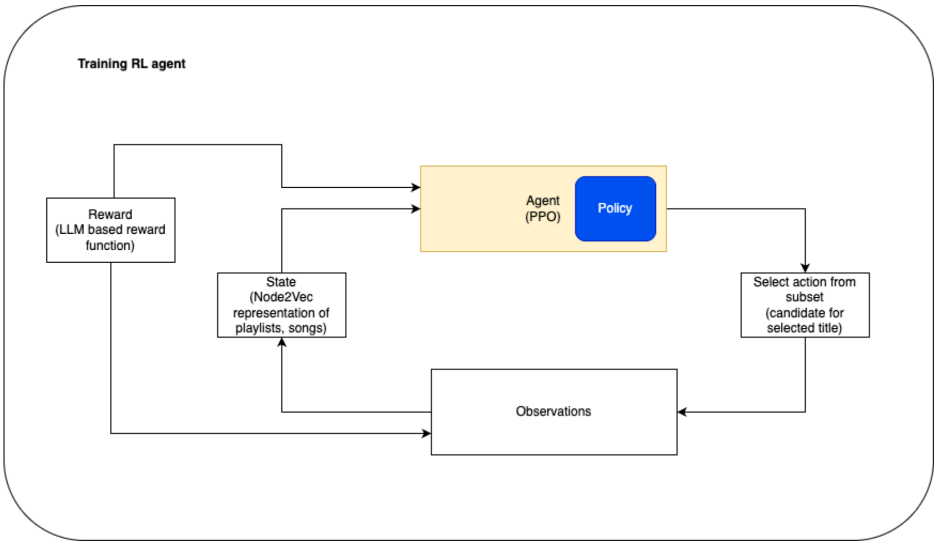

Training RL agent
Reward (LLM based reward function)
Agent (PPO)
Policy
State (Node2Vec representation of playlists, songs)
Select action from subset (candidate for selected title)
Observations

Figure 4.2: Overview of the training the RL agent for playlist generation

Table 4.1: Distribution of Clip Metadata

Song Mood

Mood	Count
NULL	795100
Normal	85883
Uplifting	38932
Melancholic	24013
Happy	10242
Sad	6207

Song Speed

Speed	Count
NULL	795815
Normal	73378
Uptempo	46910
Relaxed	33106
Ballad	8119
Extra Fast	3049

Song Urgency

Urgency	Count
Non-relevant	846926
Recognizable	82790
Classic Hit	15159
Recurrent Hit	9570
Hit	5932

process to create dense vector representations for the songs and other metadata entities.:

1. Construction of a Heterogeneous Collaborative Knowledge Graph (CKG) to model the rich network of connections between musical entities.

2. Application of a weighted Node2Vec algorithm to learn low-dimensional embeddings from the graph structure.

Constructing the Collaborative Knowledge Graph: The foundation of our song representation is a heterogeneous Collaborative Knowledge Graph (CKG) designed to model the rich network of connections between musical entities. The primary nodes in this graph are the songs themselves (represented as videos), which are described by a set of factual metadata nodes, including their associated artists, genres, subgenres, and decades. To incorporate high-level thematic organization, the graph also includes playlist nodes, representing collections curated by human experts. Finally, to capture user behavior, the graph is populated with anonymized user nodes.

The relationships between these entities are defined by a multi-source edge structure. The semantic backbone of the graph is formed by metadata-driven edges, which create factual links such as song-has-artist or song-has-genre. This structure is then enriched with expert-curation edges (playlist-contains-song), which encode the invaluable knowledge of professional curators regarding which songs belong together. The final layer incorporates collaborative filtering edges, such as user-streams-song and user-likes-song. Adding this user interaction data transforms the graph into a true CKG, allowing the model to learn latent user preferences and similarities that are not explicitly captured in the metadata alone. This multi-faceted structure enables the discovery of complex relationships between songs via multi-hop connections, providing a rich, contextual foundation for our recommendation agent.

Multi-modal Enhancement with Audio Similarity While the CKG effectively captures semantic and collaborative signals, it may not effectively capture the acoustic similarities between songs. To address this limitation, we enhanced the graph with a new modality of information derived directly from the songs' audio content. By adding direct song-song connections based on acoustic similarity, we allow the Node2Vec random walks to traverse directly between sonically similar tracks, even if they do not share common artists or genres. This results in a shortcut for discovering fine-grained acoustic relationships. The implementation proceeded by first generating a vector representation for each song's log-mel spectrogram using the pre-trained MULE audio embedding mode. Using the Faiss library, a k-nearest neighbor search (where k=10) was then performed in this audio embedding space for each song. Finally, new edges of type audio_similar were added to the CKG between each song and its ten nearest neighbors. To ensure these new acoustic signals supplemented rather than dominated the existing graph structure, their influence was carefully controlled: the edge weights, derived from the cosine distances, were scaled to a constrained range of [0, 0.125], thereby balancing the multi-modal information within the graph. This specific range was determined by the experiments in our baseline paper [96], which demonstrated on a playlist completion task that a smaller audio edge weight (0.25) yielded superior performance over other weights (0 and 0.5), establishing that a lower weight is optimal for balancing the multi-modal signals within the graph.

Generating Embeddings with Weighted Node2Vec With the multi-modal CKG fully constructed, we applied the Node2Vec algorithm [43] to learn a 128-dimensional embedding for each node. Similar to the baseline paper, our implementation utilized the computationally efficient fastnode2vec library, which was specifically configured to respect the pre-defined edge weights (weighted=True). This ensures that the probability of a random walk traversing an edge is proportional to the edge's importance, whether derived from collaborative signals, expert curation, or acoustic similarity. To ensure the robustness of our embeddings, the Node2Vec hyperparameters were determined via a systematic grid search. The final configuration used for generating the embeddings for our RL agent was:

- Embedding Dimensions (d): 128
- Walk Length: 20
- Context Window Size: 5
- Return Parameter (p): 0.5 (encourages broader exploration)
- In-out Parameter (q): 0.5 (encourages staying within a local neighborhood)
- Training Epochs: 200

To ensure the integrity of our evaluation process, all nodes for the playlists designated for the hold-out test set were entirely removed from the graph before the Node2Vec training process began. This strictly prevents any information about the evaluation data from leaking into the learned song embeddings, ensuring that our downstream evaluation of the RL agent is fair and unbiased.

The output of this process was a dictionary mapping each song ID to a multi-modal, 128-dimensional vector. These embeddings serve as the foundational state representation for the RL agent, equipping it with a multi-modal understanding of the music catalog.

Inclusion of Popularity Metrics

A major limitation of the data used for our research is the exclusion of external sources like Wikipedia, Billboard and Spotify charts, which the experts leverage to determine song and artist popularity, globally or contextually (e.g. for a specific genre or decade). Additionally, the available popularity metric in the song metadata was categorical, corresponding to 5 categories : “Non-relevant“, “Recognizable“, “Classic Hit“, “Recurrent Hit“ “Hit“, which was not representative enough to differentiate between the songs' in terms of popularity. Similarly, there was no readily available metric to compute artist popularity. To overcome these limitation, we designed our own metrics for artist and song popularity.

For computing song popularity, we used the play counts for a given song. Additionally to determine

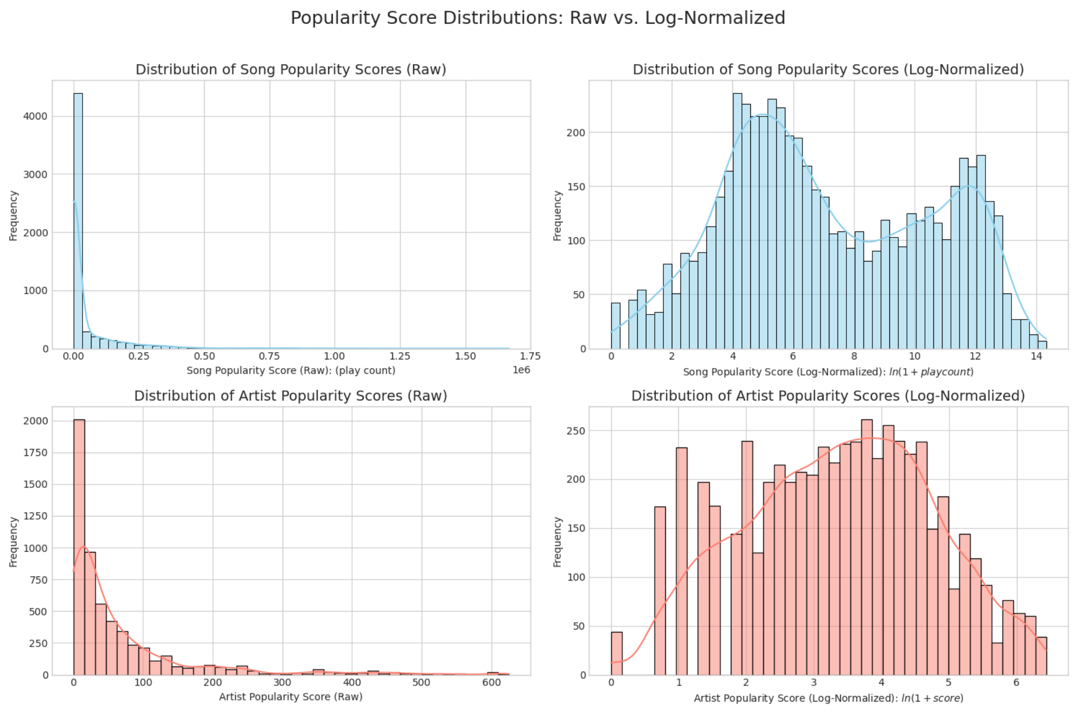


Figure 4.3: Distributions of the Song and Artist Popularity Scores.

artist popularity, we assigned a weighted score based on the popularity categories of the songs featuring the artist. Both metrics resulted in a highly skewed, long-tail distribution, where a small number of songs and artists have extremely high popularity scores (above a million for a viral hit), while the vast majority have very low scores (closer to 0). Such distributions in the data can effect the agent's learning, as the extreme outliers can disproportionately influence the model's parameters, often overlooking the significance of less popular items. To mitigate this imbalance, we applied a logarithmic transformation to compress the data's scale. We computed the natural logarithm of 1 plus the raw score:

$$y = ln(1 + score) \tag{4.1}$$

This transformation (as shown in Figure 4.3), helped in reducing the differences caused by high value outliers and transformed the skewed distribution into a more Gaussian distribution. Through this normalization, we ensure that the popularity features contribute to the model's performance without being dominated by a few highly popular tracks.

Multi-Component State Vector

o enable our agent to make informed decisions throughout the playlist generation process, we design a rich, multi-component state vector. At each step t, the state s_t provided to the agent is a dictionary comprising several components, each designed to capture a particular features of the playlist constructed so far. These components are combined using a MultiInputPolicy mechanism and then provided as input for the neural network. At each step t, the state s_t provided to the agent is a dictionary containing multiple components, each designed to capture a distinct aspect of the playlist constructed so far:

- **Average embedding of songs in the playlist:** The mean of all Node2Vec embeddings for tracks selected so far. This gives the policy an estimate of the playlist's overall thematic coherence.
- **Full playlist embeddings:** A stacked sequence of the Node2Vec vectors for each chosen track, allowing the agent to learn learn higher-order transitions and long-range dependencies [90].
- **History embeddings):** A sliding window over the embeddings of the last $\ell = 5$ tracks to help the agent focus on recent transitions and local similarity, reflecting the finding that local audio similarity strongly predicts user satisfaction [51].

- **Last track embedding:** The embedding of the most recent song, allowing the policy to explicitly condition on the immediate predecessor, supporting smoother transitions [39].
- **Popularity statistics:** Summary statistics (mean, std) of the log-normalized popularity scores for all songs and artists in the current playlist, enabling the agent to balance familiarity and novelty, thus mirroring the diversity–popularity trade-off in [39]

Our agent's decision-making relies on the above diverse set of information, presented as a dictionary of different data types. To handle this, our policy uses a "multi-input" neural network architecture. This means that instead of trying to process all the different pieces of information at once in a single stream, the policy breaks down the observation dictionary.

For each distinct type of information, like our song embeddings or popularity scores, the policy creates a separate, specialized "sub-network". These sub-networks then work in parallel to extract relevant features from their specific data, which are then concatenated together into a single feature vector. The sub-networks for larger vectors like sequences of song embeddings use RNNs to understand patterns over time, while smaller dimensional data like popularity scores go through feedforward networks. This combined vector then becomes the main input for the rest of the policy's neural network. Together, the vectors gives the RL agent both overall context (average embedding, full sequence) and local context (recent history, last embedding), as well as quantitative signals of popularity and progression.

Reducing the Action Space - Invalid Action Masking

Similar to previous works on RL-based playlist generation[39, 7], our action space is represented by song ids where the songs ids are discrete values mapped to each songs in the playlist. To train the agent, the first challenge is dealing with the large action space problem 2, as the number of songs available for playlist generation can be large.

To address this issue, instead of the agent having to explore the entire song database in each episode, we employ action masking. This approach significantly reduces the number of songs the agent needs to consider at each step, making the selection process faster and more efficient. During our interviews with the experts, we tried to understand if any filter was used at the beginning of curating their playlists, for which we got that the experts used an internal database tool to filter out songs matching the targeted theme based on the songs' metadata . For example, while curating playlists like '90s pop' or 'Hits now', the songs belonging to that specific decade (1990), genre ('Pop'), year (2025), popularity ('Hit') would be filtered. This makes the process of handpicking songs easier for the expert. Following a similar approach while training our agent, we only provide the agent with the songs which match the given theme profile as the unmasked actions. This serves as a filter to the agent, and a way for the agent to navigate through large discrete action spaces. Additionally, we mask out the songs already present in the playlist to avoid repetition within the playlist.

4.2.2. Agent Training and Optimization

To optimize our playlist-generation policy, we employ the Maskable Proximal Policy Optimization (MaskablePPO) algorithm [66], which extends the standard PPO framework to support large discrete action spaces with dynamic masking. We use a `MultiInputPolicy` to handle our heterogeneous state representation.

Hyperparameter Tuning

The performance of a RL agent is highly sensitive to the choice of its hyperparameters. To ensure that our results are robust and not merely an artifact of a specific configuration, a systematic hyperparameter optimization process was conducted. This process aimed to identify a near-optimal set of hyperparameters for the PPO algorithm, which serves as the foundation for our learning agent.

To maintain experimental integrity and ensure a fair comparison across all models, we adopted train our agent on a fixed set of hyperparameters, determined through hyperparameter tuning performed on the manual reward RL agent, similar to the approach presented by Xie et al [26]. The manual agent serves as our primary baseline, and by optimizing its configuration, we establish a consistent foundation. The optimal set of hyperparameters discovered during this process was then fixed and used for the training of all subsequent models, including those guided by the LLM-derived reward functions. This ensures that any observed differences in performance can be attributed to the quality of the reward signal itself, rather than variations in the agent's underlying training configuration.

We utilized the Optuna framework [106], an open-source hyperparameter optimization library, to automate the search process. Optuna employs efficient sampling and pruning algorithms to explore the hyperparameter space and find optimal values. The optimization was structured as a series of 30 trials. In each trial, a set of hyperparameters was sampled from a predefined search space, and an agent was trained for 200000 timesteps using these parameters. The objective function for each trial was to maximize the mean reward obtained by the agent over five evaluation episodes in a separate, unseen environment.

Based on the provided implementation, the following PPO hyperparameters were tuned, with their respective search spaces defined as follows:

1. **Learning Rate (learning_rate)**: Sampled from a logarithmic scale between 1e-5 and 1e-3. This parameter controls the step size of the optimizer.
2. **Discount Factor (gamma)**: Uniformly sampled from the range [0.90, 0.999]. This determines the importance of future rewards.
3. **Number of Steps (n_steps)**: Categorically chosen from [512, 1024, 2048]. This is the number of steps to run for each environment per update.
4. **Batch Size (batch_size)**: Categorically chosen from [32, 64, 128]. This is the size of the mini-batch used for each gradient update.
5. **Number of Epochs (n_epochs)**: An integer sampled from [5, 20]. This is the number of times the agent iterates over the collected data during an update.
6. **Clip Range (clip_range)**: Uniformly sampled from the range [0.1, 0.3]. This is a core PPO parameter that clips the policy update to prevent excessively large changes.

Upon completion of the 30 trials, the set of hyperparameters that yielded the highest mean reward was identified as the optimal configuration. This single, optimized configuration was subsequently used for all experiments presented in this thesis, ensuring consistency and comparability across all tested reward functions. The results from the hyperparameter tuning are presented in Appendix C

Training Protocol

To ensure the statistical reliability and reproducibility of our findings, a rigorous training protocol was established and followed for all experiments. The stochastic nature of RL, arising from random weight initialization and probabilistic action selection, necessitates multiple training runs to form a credible assessment of an agent's performance. Consequently, the performance of an agent from a single training run can exhibit high variance and may not be representative of the algorithm's true capabilities. A single successful run could be an outlier, just as a single poor run could unfairly represent the model's potential.

For each experimental condition (i.e., each reward function), we conducted five complete training runs, each initiated with a different random seed. All agents were trained for a total of 5 million timesteps using the optimized hyperparameters identified in Section 4.2.2.

5 Experimental Setup

The goal of our experiments is to evaluate how effectively our proposed system captures expert preferences. This involves comparing playlists generated by our expert-driven models against the baseline models with respect to reference playlists curated by human experts. The evaluation is designed to assess not only the accuracy of song selection but also the overall characteristics of the generated playlist, in terms of sequencing and attributes of the tracks picked, ensuring the models can generate playlists that can align with the standards of a professional curator.

5.1. Models compared

To assess the performance of our approach, we compared the performance of RL agents trained on different reward functions (LLM-generated and manually crafted), and the performance of the Node2Vec model introduced in Section 3.2. The state and action space representations in the training environment are held constant across all RL agents to ensure a fair comparison focused on the efficacy of the reward functions.

The models evaluated are as follows:

- **Node2Vec Similarity Baseline (*N2V-S*):** This non-RL baseline generates playlists by sequentially selecting songs with the highest cosine similarity to the current playlist's average embedding. The embeddings are derived from the Node2Vec model used for the RL agent's state representation, as detailed in Section 4.2.1. This model serves as a baseline for assessing the effect of the reward function on RL agent performance, given that it uses the same Node2Vec embeddings for state representation.
- **Manually-Crafted Reward RL Agent (RL-M):** The RL-M agent is trained using a hand-crafted reward function which integrates four quantifiable aspects of playlist quality, inline with the manually designed reward function of Shih et al. [39]. The reward factors were redefined to reflect the researchers' insights from the expert interviews and to align with the used data sources (Node2Vec embeddings, raw popularity scores), which constitute the state and action spaces of our agents. The reward components are calculated as follows:
  - Coherence (F): the mean cosine similarity between the embeddings of consecutive tracks, with negative similarity values decreasing the flow score.
  - Popularity (P): the average of each track's popularity score after normalization across the dataset bounds.
  - Novelty (Q): the mean of inverse of the popularity scores of the songs added, to expose the listeners to lesser known tracks (or "hidden gems") that they might not otherwise encounter.
  - Diversity (D): the proportion of unique artists within the playlist, to encourage songs from multiple artists, a criteria identified during the expert interviews as well as in [39].

  These components are combined into a single scalar reward:

$$R = 0.3\,F + 0.3\,P + 0.2\,Q + 0.2\,D \tag{5.1}$$

where the reward factor weights (0.3,0.3,0.2, and 0.2) were determined manually based on the inputs from the experts. This agent acts as a RL baseline to determine whether the LLM-derived reward functions provide an advantage over a more direct, human-crafted design.

- **LLM-Generated Reward from Summarized Transcripts (RL-LLM-S):** This represents our proposed RL agent trained on the LLM-generated reward function from Pipeline A (4.1.2). The objective of this model is to assess the impact that combining and condensing the expert interviews can have on the reward functions generated by the LLMs and subsequently on the performance of the trained RL agent.
- **LLM-Generated Reward from Raw Transcripts (RL-LLM-R):** This model explores the impact of Pipeline B (4.1.2). It follows a similar structure as the *RL-LLM-S* agent, with the exception that of the input source to the LLM for reward function generation. By comparing the performance of RL-LLM-R to RL-LLM-S, we evaluate the effect of introducing the additional summarization and refinement step in the methodology pipeline.

5.1.1. LLM Models

For the LLM-based reward generation, we compare the effectiveness of three different LLMs: OpenAI's GPT-4 , Google's Gemini Pro, and Anthropic's Claude introduced in Section 2.3.1. This comparative analysis allows us to evaluate if the choice of LLM model can influence the quality and alignment of the generated reward functions. For each input type (from Pipeline A and Pipeline B), the models were prompted 5 times to generate 5 different reward functions, which enabled us to account for the variation within the outputs generated by each model.

5.2. Dataset and Evaluation Procedure

One of the main phases in the experimental design involved understanding the XITE dataset. While analyzing the expert-curated playlists titles, we noted that out of 789 playlists that exist, the titles of 453 playlists were based on songs' metadata (eg: "90s Pop", "80s Rock"). For the scope of our evaluation, we decided to train our RL agents on playlist titles generated from the songs' metadata, specifically of the format "decade", "genre", and "decade+genre". Additionally, for evaluating our models, we test it on expert-curated playlists which followed the format of "decade+genre". Additionally, the song pool contains 5755 songs that either adhere to the theme of training and evaluation playlist titles or are present in the expert curated test playlists. The agent was trained on 145 such playlist titles and evaluated against 18 expert-curated playlists.

Another important thing to note is that the expert-curated playlists on the platform vary in length from 10 to over 4000 tracks per playlist. We fixed the length of the generated playlists as 20 songs during training and evaluation. This length was chosen as it corresponds to the average user streaming session for the expert-curated playlist titles over the past year, allowing us to focus our analysis on a realistic use case.

5.2.1. XITE Bucketing system for Song Sequencing

A key challenge in evaluating playlist generation systems is that there is not one single "correct" sequence for a given playlist. Within XITE, the experts themselves use a "bucketing" system for partial ordering, where songs within a bucket are interchangeable, but the buckets themselves have a defined order. When a user plays a playlist multiple times, the ordering of the songs can be different each time. This indicates that multiple sequences exist that meet the experts' ordering criteria. To account for this variability, we generated 1000 different sequences for each of the 18 test playlists by repeatedly querying the platform's API. These 1000 sequences serve as our ground truth reference set. Thus the evaluation is performed by comparing the sequences generated by the models for a given title to each of the sequences present in the expert-curated reference set for that specific title. Given that the length of the model-generated playlists has been fixed to 20 tracks, we consider the top 20 songs from any expert playlist sequence. This also implies that there is variability in the song selection across the 1000 valid 20-song sequences for a playlist, meaning they do not all contain the exact same set of tracks. Figure 5.1 gives an overview of the sequences generation and playlists comparison.The figure shows the evaluation done for one of the test playlists of length 5. In our experiments, we follow these for each test playlist with a length of 20 tracks.

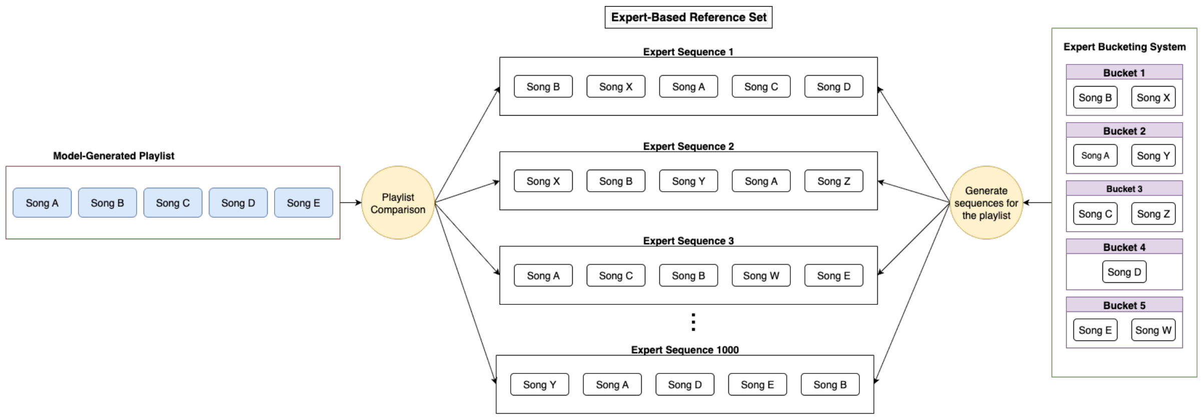


Figure 5.1: An overview of the evaluation methodology based on the expert bucketing system. Our ground truth is generated from the expert curated playlist using the bucketing system (on the far right), where songs within each ordered bucket are interchangeable. The evaluation process involves a one-to-many comparison, assessing the playlist generated by the model(on the left) against each of the 1000 sequences in the experts' reference set (in the middle).

5.2.2. Playlist Completion Task

The core method used in our evaluation is the playlist completion task, as proposed by [52]. Given a number (N) of seed tracks, the model needs to select and order the remaining (20-N) tracks for the playlist from the playlist song pool, where 20 is the length of the playlist. The song pool is filtered using the action mask introduced in Section 4.2.1, which filters out songs that do not match the theme of the given playlist. We evaluate the models on the playlist completion task for two distinct scenarios:

1. **Seedless Generation (N=0):** In this scenario, the models are provided with the playlist title (e.g., "90s Pop") and no other context, to generate a complete 20-song playlist. This setup evaluates the models' ability to address the cold-start problem (introduced in Section 2.1.3), as it relies entirely on its learned policy without any initial context. When evaluating the N2V-S model, we provide the embedding of the playlist title, which is obtained by taking the mean of the embeddings of the components for the playlist title. For example, for the playlist "90s Pop", the title embeddings is computed by averaging the Node2Vec embeddings of the decade "1990" and the genre "Pop". This embedding serves as a starting point for the model, with the model initially picking the song with embedding closest to the playlist title embedding from the song pool. The RL agents on the other hand, use their learned policies to generate playlist from the song pool.

2. **Seeded Generation (N=1):** Here, the agent is provided with the playlist title and the first song from the expert-curated sequence to which it is compared. It then generates the remaining 19 songs. This scenario tests the model's ability to contextually continue a playlist, adapting its choices based on an initial track. For the N2V-S model, the Node2Vec embedding of the first song from the sequence is provided to the model, which then determines the successive tracks of the playlist based on the cosine similarity. For evaluating the RL agents, information pertaining to the first track (Node2Vec embedding and popularity scores) are incorporated into the state representation, based on the the agent picks the successive tracks.

5.3. Evaluation Metrics

To ensure a comprehensive evaluation, we split our evaluation into two categories of metrics. The first category, *Recommendation Quality Metrics*, uses standard offline metrics (Precision@k, Recall@k and NDCG@k) to measure the performance of our models against the expert-curated ground truth. These metrics assess the accuracy of song selection and ranking. We introduce a second category of metrics, *Reward Component Metrics*, to evaluate the characteristics of the generated playlists according to the curatorial factors derived from expert interviews. The two metric categories allow us to evaluate the generated playlists in terms of specific tracks picked as well as the “profile“ of the generated playlists.

5.3.1. Recommendation Quality Metrics

The following metrics are used to compare the generated playlists against the expert-curated reference sets. For each model and playlist title, the metrics are computed for the models’ generated playlist against the sequences in the reference set and then averaged across the sequences of that title. The scores obtained determine the overlap between the song set as well as the sequencing of songs. We focus primarily on k values of 10 and 20, as it relates directly to our fixed playlist length and typical user behavior. We select k=20 to assess accuracy and ordering across the entire 20-track playlist and k=10 to focus on the first half, since listener engagement at the beginning of a playlist can influence session continuation.

1. **Precision@k:** Measures the proportion of relevant songs (i.e., songs present in the expert-curated playlist) among the top k songs recommended by the model. In our setup, this helps evaluate how accurately the model is able to select expert-relevant tracks. Higher precision indicates better alignment with the expert’s song selection.

$$\text{Precision@}k \;=\; \frac{|\{\text{recommended}_1^k\} \cap \{\text{expert playlist}\}|}{k} \tag{5.2}$$

   This measures the fraction of the top-k tracks that appear in the expert’s playlist.

2. **Recall@k:** Measures the proportion of relevant songs that were retrieved in the top k recommendations out of the total relevant songs in the expert playlist. In this context, Recall@k helps assess how well the model captures the expert’s choices within its top-k predictions.

$$\text{Recall@}k \;=\; \frac{|\{\text{recommended}_1^k\} \cap \{\text{expert playlist}\}|}{|\{\text{expert playlist}\}|} \tag{5.3}$$

   This measures the fraction of the expert’s tracks that are recovered among the top-k.

3. **NDCG@k (Normalized Discounted Cumulative Gain):** Captures both the relevance and the position of songs in the recommended list. It gives higher weight to relevant songs appearing earlier in the sequence. This metric focuses on the ordering of tracks. We consider ndcg@k as a primary indicator, as it evaluates both the relevance of the selected songs and their position in the sequence. A higher NDCG@k score implies that the model not only selects appropriate songs but can also order them in a manner consistent with expert-curated playlists. To compute NDCG@k, we first define the Discounted Cumulative Gain (DCG):

$$\mathrm{DCG@}k \;=\; \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}, \tag{5.4}$$

   where

$$rel_i = \begin{cases} 1, & \text{if the } i\text{-th recommended track is in the expert playlist,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

   Then normalize by the ideal DCG (IDCG), which is the DCG of a perfect ranking (all k expert tracks first):

$$\mathrm{IDCG@}k \;=\; \sum_{i=1}^{\min(k,|\text{expert playlist}|)} \frac{1}{\log_2(i+1)}. \tag{5.6}$$

   Finally,

$$\mathrm{NDCG@}k \;=\; \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}. \tag{5.7}$$

   This captures both whether the expert selected tracks appear in the top-k and how early they appear. This is relevant since we want the users to be engaged with the playlists from the very beginning.

These metrics allow us to evaluate both the selection quality and sequencing capability of our models, providing insights into how well they approximate expert-level playlist curation.

5.3.2. Reward Component Analysis

In addition to the primary recommendation metrics, we evaluate our models using a secondary set of metrics relating to the expert-oriented characteristics of the generated playlists. While objective measures like Precision, Recall and NDCG are valuable for assessing recommendation accuracy against a ground-truth dataset, they fail to capture the qualitative aspects that define expert curation. The goal of this research is not merely to recommend relevant songs, but to generate playlists that reflect the expert principles like “flow“, “strong start“, and balance between “popularity“, and “novelty“. Consequently, the comparison of the models based on these metrics allows us to assess which models produces playlists that are most characteristic of the experts’ own creations.

From Expert Intuition to Computable Profiles

One of the challenges in determining the characteristic of the playlist is defining curatorial concepts like “flow“, “diversity“, and “novelty“ as quantitative metrics. These factors do not have a universal definition, and their interpretation can vary between experts and contexts. As an example, one of the experts explained flow as ‘...for an Indie playlist, you don’t really need to make a difference in tempo, so I’d say maintain a regular tempo, and you don’t need to build up the energy“, which would indicate that higher cosine similarity between consecutive songs would indicate a better flow. However, the definition of flow was presented a bit differently by another expert, *“I made six buckets and made the first one such that it would start with nothing too fast or nothing too slow but also something very recognizable, and then went into something maybe a bit heavier for the second bucket and then back down a bit slower and then build it up and down. So then when you’re listening to it, the tempo is flowing up and down.“*, suggesting that while the cosine similarity between consecutive songs should not be too low, it should also be maintained below a certain threshold.

The quantitative definitions provided for each metric, therefore, is our attempt at operationalization of these concepts. Operationalization is the process of defining abstract and sometimes ambiguous concepts in quantifiable terms and is a widely practiced in the field of recommender systems [107, 108] for measuring fairness, diversity, etc. The hypothesis is not that our definition of the concepts are the ground truth, but that an agent trained to optimize on reward functions derived from expert curation principles should learn a policy that generates playlists with characteristics similar to those of the experts. The selection of our five reward component metrics, flow, popularity, start power, diversity, and novelty was based on a manual analysis of our interviews with the experts, representing the most consistently articulated principles in their explanations. Furthermore, the objective of this comparison is not to declare that a higher score in a metric like “flow“ is better. Instead, we treat the collection of expert-curated, ground-truth playlists as the benchmark that defines a target “expert profile“. For these metrics, a model’s effectiveness is measure by how the close the scores obtained by each model are to the expert sequences.

Reward Component Metrics

The evaluation involves scoring each generated playlist sequence from the models, as well as the expert reference sequences, against the reward component metrics. For computing the expert profiles for a given playlist title, we average out the scores obtained by the playlists’ corresponding sequences in the reference set. Given, a generated playlist be a sequence of L songs, S=(s_1, s_2 ,...,s_L), the reward component metrics are defined as:

- **Flow Score (F):** Measured as the sum of the cosine similarities between 2 consecutive songs’ embeddings ($v(s_i)$) to its context. For the first song, the context is the playlist title embedding (v_T). For subsequent songs, it is the embedding of the previous song ($v(s_{i-1})$). The score is scaled to a $[0, 1]$ range.

$$F(S) = \sum_{i=1}^{L} \frac{1}{2} \left(\mathrm{sim}(v(s_i), \begin{cases} v_T & \text{if } i = 1 \\ v(s_{i-1}) & \text{if } i > 1 \end{cases}) + 1 \right) \tag{5.8}$$

  The concept of flow appeared several times across the interviews, particularly while describing the ordering of the songs. It was captured in the summarized document obtained from Pipeline A as well. Flow happened to be one of the toughest concepts for the experts to articulate, with many further describing it with terms like “vibes“, “feel“, and “mood“: *“Since I’m very familiar with* *hip and R&B, I already have the vibe and the flow in mind when curating for such genres"*. As expected, there was not one single definition for flow: *"..the flow would include listening and hearing some of the biggest hits followed by some of the smaller songs that didn't do so well"* or as another expert puts it, *"..if we go from Indie to EDM I'm fine with that because I like both of those genres. But yeah, it really flows, if it's in the same kind of ballpark... if it's in the same kind of vibe..."*.

Our definition of flow was motivated based on the description provided by a majority of the experts as the smooth sequencing of songs, *"I think my favorite way of like separating energy is to build it up and then build it back down again. I think it's really jarring to go from a really powerful song to a really chill song."*, indicating a tendency to pick songs which would be closer to each other in the node2vec embedding space

- **Popularity (P):** The combined popularity score, calculated as the sum of the log-normalized song popularity score (p_{song}) and the log-normalized artist popularity score (p_{artist}) for the tracks added to the playlist.

$$P(S) = \sum_{i=1}^{L} (p_{\text{song}}(s_i) + p_{\text{artist}}(s_i)) \tag{5.9}$$

  This metric reflects the strategic use of well-known, popular songs ("hits") to ensure the playlist is recognizable and engaging for a broad audience, *"The base of the playlists are rooted in popular songs, I think it may be like a 95 to 5 ratio for popular to unpopular songs."*. The consensus on what the experts considered popular varied slightly, *"If you want to dig in to see what's popular, you can see the number of streams that a song has and that helps."*, while another expert factored in artist popularity *"The song's popularity is the most important because you want people to keep it on and sing along. But with big artists like Beyoncé, I don't think it really matters what song is played."*

- **Starting Power (S):** The total starting power score is the sum of bonuses awarded for adding highly popular songs within the first N_{start} tracks (where $N_{\text{start}} = 3$), reflecting the "start strong" heuristic.

$$S(S) = \sum_{i=1}^{N_{\text{start}}} \begin{cases} 1.0 & \text{if } p_{\text{song}}(s_i) \geq 0.75 \\ 0.0 & \text{otherwise} \end{cases} \tag{5.10}$$

  Here, we consider only the songs' popularity, since the experts generally are more concerned on starting the playlists with trending hits, instead of focusing on the artist's popularity, *"You want to start off with the biggest hits, at least like three big hits in a row."*

- **Diversity (D):** The total diversity score is the sum of step-wise scores that reward adding new artists and penalize repetition. At each step i, a raw score $r_{\text{div}}(s_i)$ is calculated, clipped to $[-1, 1]$, and scaled.

$$D(S) = \sum_{i=1}^{L} \frac{1}{2} (\text{clip}(r_{\text{div}}(s_i), -1, 1) + 1) \tag{5.11}$$

  A majority of the experts interviewed emphasized on the importance of representing multiple artists within the same playlist, *"I would definitely try to not have too much artist repetition."*

- **Novelty (N):** The total novelty score for the playlist is the sum of the inverse log-normalized song popularity scores for all tracks.

$$N(S) = \sum_{i=1}^{L} (1 - p_{\text{song}}(s_i)) \tag{5.12}$$

  This metric captures the goal of balancing familiar hits with lesser-known tracks or "hidden gems" to help listeners discover new music, *"I think that's my key main driver, to sort of help people discover new, but it could also be old music."*

Comparing the average scores for these components across the different models and the expert-curated ground truth provides insight into the behavioral alignment of each agent. We expect the expert-driven RL-based model to produce playlists whose characteristic scores for flow, diversity, and novelty closely mirror those of the expert reference set, thus depicting a better understanding of expert curation beyond simple item selection.

5.4. Statistical Significance Testing

To ensure the robustness of our findings, a statistical significance analysis was conducted to validate that observed performance differences between models were not attributable to random chance. The objective is to determine if the superior performance of one model over another is statistically meaningful.

Given that the performance scores may not follow a normal distribution, the non-parametric Wilcoxon signed-rank test was selected as the primary statistical tool. This test is appropriate for our setup, as it directly compares the paired scores (one for each playlist) between two models. It works by ranking the differences between the paired scores and evaluates if one model consistently outperforms the other across the entire test set. For each evaluation metric, pairwise comparisons were performed between all models. The significance testing was structured to address the specific comparisons in each research question:

- For RQ1 (Comparing Proposed Approaches): To evaluate the impact of the input processing pipeline, the averaged scores of the RL-LLM-S and RL-LLM-R models (for each LLM) were directly compared using the paired Wilcoxon test. Additionally, we
- For RQ2 (Comparing Against Baselines): To evaluate our proposed models against the baselines, the paired Wilcoxon test was used for all comparisons: RL-LLM-S vs. N2V-S, RL-LLM-S vs. RL-M, RL-LLM-R vs. N2V-S, and RL-LLM-R vs. RL-M.

The null hypothesis (H0) for each test was that the median difference in performance scores between the two models is zero. A significance level (α) of 0.05 was used as the threshold.

Furthermore, to address the heightened risk of Type I errors (false positives) when performing multiple statistical comparisons on the same dataset, we applied a Benjamini-Hochberg (BH) correction, which controls the False Discovery Rate (FDR), which is the expected proportion of false positives among all declared significant findings. A comparison between two models was considered statistically significant only if its BH-adjusted p-value was less than 0.05. This rigorous testing provides a high degree of confidence in the results presented in Chapter 6

6 Results

This chapter presents the experimental results and qualitative analysis conducted to address the research questions of this thesis. It is structured as follows: Section 6.1 addresses RQ1, focusing on the impact that the choice of the different input approaches and LLMs has on performance. Section 6.2 addresses RQ2 by comparing our proposed models' performance against the N2V-S and RL-M baselines. Each section combines quantitative metrics with qualitative insights to provide a comprehensive analysis of model performance.

6.1. RQ1: Impact of Transcript Processing and LLM Choice

This section examines the impact of the input processing pipeline (RL-LLM-S vs. RL-LLM-R) and the choice of LLM on the performance of the resulting RL agents. The analysis begins with a quantitative comparison using the evaluation metrics defined in Chapter 5, followed by a qualitative analysis of the generated reward functions.

6.1.1. Quantitative Analysis

To measure the effect of these design choices, the performance of each model configuration is assessed using recommendation quality and reward component metrics. The final score reported for any given metric is the average performance across all playlists in the test set. This is calculated as shown in Equation 6.1:

$$\text{Score}(M, K) = \frac{1}{|P_{\text{test}}|} \sum_{p \in P_{\text{test}}} \text{score}_p(M, K) \tag{6.1}$$

where:

- Score(M,K) is the final score for model M on metric K.
- P_{test} is the set of all playlists in the test set.
- $|P_{test}|$ is the total number of playlists in the test set.
- $score_p(M, K)$ is the raw score achieved by model M on metric K for a single playlist p.

Recommendation Quality Metrics

The analysis uses precision@k and recall@k to measure the accuracy of song selection and ndcg@k to evaluate the quality of the song ranking. We reported scores for both k values of 10 and 20 to distinguish performance on the first half of the playlist from the full 20-song playlist. Tables 6.1 and 6.2 present the scores achieved by the models in the seedless and seeded playlist completion tasks.

Across all models and pipelines, precision, recall, and nDCG scores remained between 0.1 and 0.3. This range is a result of our evaluation framework, which defines the ground truth for a playlist as the top 20 tracks from each 1,000 sampled expert sequences. The significant variance in the length (10-4,000 tracks) and internal ordering of the source playlists means that many valid recommendations may not be present or may have an inconsistent rank within the truncated evaluation set. Under these conditions, scores in this range are expected and interpreted as representing substantial alignment with expert choices.

To validate our findings, we performed statistical significance testing. For each metric, a Wilcoxon signed-rank test was conducted for each pairwise model comparison, and the resulting p-values were adjusted using the Benjamini-Hochberg procedure to control the false discovery rate at a significance level of α=0.05.

Seedless Playlist Generation (N=0) In the seedless scenario, the performance differences between the summary-based (RL-LLM-S) and raw transcript-based (RL-LLM-R) pipelines vary considerably depending on the underlying LLM, as shown in Table 6.1.

For the Claude-based models, the RL-LLM-S pipeline demonstrated a marginal yet consistent advantage. At the k=10 cutoff, its precision@10 score of 0.2714 indicates that it placed, on average, 2.7 expert-selected tracks in its top ten, compared to 2.6 for the RL-LLM-R model (precision@10 of 0.2589). The higher recall@10 and nDCG@10 scores further suggest that the summary-based approach not only covered a slightly larger portion of the expert's valid continuations but also ranked them more effectively. This trend is more apparent at the k=20 level. The precision@20 score of 0.2710 for RL-LLM-S, compared to 0.2536 for RL-LLM-R, translates to a difference of nearly one additional expert-approved track for every three playlists generated. This indicates a more consistent long-term recommendation strategy from the summary-based model. Furthermore, the higher nDCG@20 score (0.2706 vs. 0.2546) further confirms that the summary-based model ranked these correct tracks more effectively, demonstrating a more consistent long-term recommendation strategy.

The GPT-based models exhibited the most substantial difference in performance between the two pipelines. The RL-LLM-S model achieved a precision@10 of 0.2026, including roughly two expert-approved tracks in its top ten. In contrast, the RL-LLM-R model's precision@10 dropped to 0.1169, corresponding to just over one correct track. The accompanying decline in the nDCG@10 score, which was nearly halved from 0.2035 to 0.1099, confirms that the raw transcript pipeline not only selected fewer correct tracks but also demonstrated a lower quality in ranking them. This pattern persists at k=20, where the RL-LLM-S model maintains its lead, while the RL-LLM-R model's performance suggests a loss of almost two expert-selected tracks per playlist.

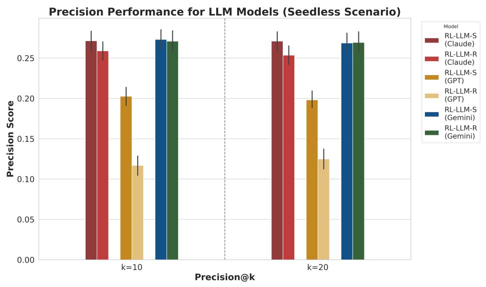
Table 6.1: Recommendation Quality Metrics for RQ1: Seedless Scenario (N=0). Bold scores indicate the better-performing approach for each LLM. Significance markers denote statistically significant differences ($\alpha = 0.05$) between the RL-LLM-S and RL-LLM-R pipelines for a given LLM, based on a Wilcoxon test with Benjamini-Hochberg correction.

	Claude		**GPT**		**Gemini**	
Metric	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R
precision@10	**0.2714**	0.2589	**0.2026**‡	0.1169	**0.2731**	0.2709
recall@10	**0.1357**	0.1294	**0.1013**‡	0.0585	**0.1365**	0.1354
ndcg@10	**0.2707**	0.2579	**0.2035**‡	0.1099	**0.2729**	0.2718
precision@20	**0.2710**‡	0.2536	**0.1981**‡	0.1249	0.2687	**0.2693**
recall@20	**0.2710**‡	0.2536	**0.1981**‡	0.1249	0.2687	**0.2693**
ndcg@20	**0.2706**‡	0.2546	**0.1998**‡	0.1179	0.2699	**0.2704**

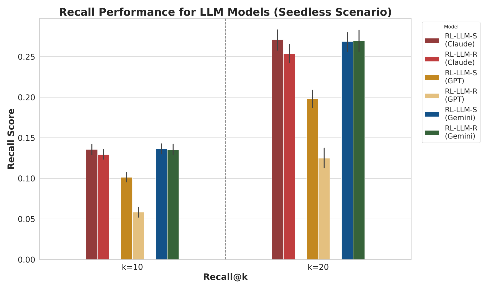
‡Significantly different from the score in the other column for the same LLM. For clarity, inter-LLM significance (e.g., Claude vs. GPT) is described in the text.

In contrast, the Gemini-based models showed minimal variation in performance regardless of the input pipeline. At k=10, the precision scores for RL-LLM-S (0.2731) and RL-LLM-R (0.2709) were nearly identical, indicating a comparable ability to select expert-approved tracks. The recall and nDCG metrics also showed negligible differences, suggesting that both pipelines produced playlists of a similar quality in terms of coverage and ranking. At k=20, precision and recall each increased with the RL-LLM-R model, marginally from 0.2687 to 0.2693, while nDCG@20 rose from 0.2699 to 0.2704, showed a marginal, likely inconsequential, increase in performance.

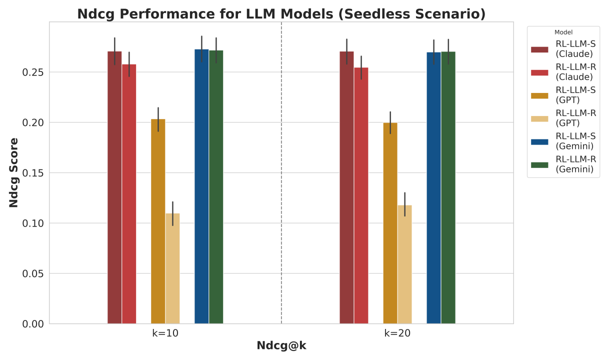
To validate these observations, statistical significance testing was conducted. For Claude, the numerical advantage of the RL-LLM-S pipeline at k=10 was found to be not statistically significant (p_{adj} > 0.05), suggesting the observed difference could be due to chance. However, at k=20, its superior performance is statistically significant across all metrics (p_{adj} > 0.05), providing strong evidence that the summarization pipeline offers a reliable advantage for generating longer playlists with this LLM. For GPT, the drop in performance when using the raw transcript pipeline was statistically significant across all metrics at both k=10 and k=20 (p_{adj} < 0.01). This confirms that the GPT model was highly sensitive to the input format and depends on the summarization process to generate an effective reward function. For Gemini, the minor differences between the two pipelines were not statistically significant for any metric, formally supporting the conclusion that its performance is robust and largely unaffected by the input processing approach.



(a) Precision



(b) Recall



(c) NDCG

Figure 6.1: Model Performance on Recommendation Quality Metrics in the Seedless Scenario, comparing RL-LLM-S and RL-LLM-R for different LLMs (Claude, GPT, Gemini).

Regarding the choice of LLMS, statistical testing confirms a clear performance hierarchy. The Claude and Gemini-based models significantly outperform the GPT-based models across both the RL-LLM-S and RL-LLM-R pipelines. The Claude and Gemini models, exhibit comparable results to each other. For instance, in the summary-based pipeline, the nDCG@10 scores for RL-LLM-S (Claude) (0.2707) and RL-LLM-S (Gemini) (0.2729) are nearly identical, and the difference between them was found to be not statistically significant. A similar lack of significant difference was observed between their RL-LLM-S counterparts. This places the Claude and Gemini models in the same performance category.

In contrast, both of these models demonstrate a statistically significant advantage over their GPT-based counterparts in all configurations (p_{adj}). While this trend is apparent in the top-10 recommendations, the performance gap is even more pronounced when evaluating the full 20-song playlists. Within the

RL-LLM-S pipeline, the precision@20 scores for Claude (0.2710) and Gemini (0.2687) indicate they successfully included, on average, 5.4 and 5.3 expert-approved tracks, respectively. This is substantially higher than the GPT model's score of 0.1981, which corresponds to finding fewer than 4 correct tracks per playlist. This suggests that for the task of generating reward functions from expert interviews, the Claude and Gemini models were more effective than those of the GPT model used in this study.

Seeded Playlist Generation (N=1) In the seeded scenario, where models began with one track and generated the remaining 19, the performance trends among pipelines and LLMs largely mirrored the seedless results, as shown in Table 6.2.

For the Claude-based models, the RL-LLM-S pipeline again yielded higher raw scores. At the k=10 cut-off, the precision@10 score decreased from 0.2712 for RL-LLM-S to 0.2585 for RL-LLM-R. However, similar to the seedless case, this difference was found to be not statistically significant. The advantage of the summary-based approach becomes more apparent and reliable for longer playlists. At k=20, the decline in performance for RL-LLM-R is statistically significant across all metrics (p_{adj} < 0.05). The drop in precision@20 and recall@20 from 0.2584 to 0.2425 translates to a practical difference of 5.2 versus 4.9 expert tracks per playlist. The significantly lower nDCG@20 score (0.2473 vs. 0.2624) further confirms that the RL-LLM-S pipeline more consistently placed expert-approved songs at higher ranks across the full playlist.

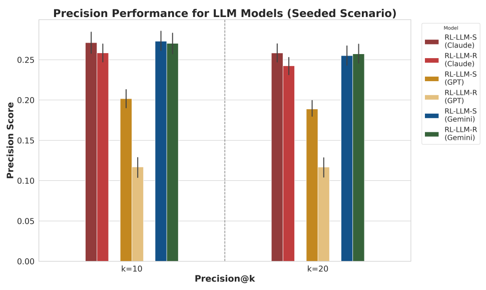
The GPT-based models once again exhibited a strong sensitivity to the input format. The sharp drop in performance from the RL-LLM-S to the RL-LLM-R pipeline was statistically significant across all metrics at both k=10 and k=20 (p_{adj} < 0.01). At k=10, the precision score fell from 0.2018 to 0.1170, meaning the raw transcript model included nearly one fewer expert-selected track in its first ten recommendations. The corresponding decline in nDCG@10 from 0.2029 to 0.1099 indicates that these correct selections were also ranked much lower. This pattern continued at k=20, where the significant drop in precision@20 from 0.1889 to 0.1168 suggests a reliable loss of more than one expert track per playlist, confirming a lower quality of recommendation with the RL-LLM-R approach.

Table 6.2: Recommendation Quality Metrics for RQ1: Seeded Scenario (N=1). Bold scores indicate the better-performing approach for each LLM. Significance markers denote statistically significant differences ($\alpha = 0.05$) between the RL-LLM-S and RL-LLM-R pipelines for a given LLM, based on a Wilcoxon test with Benjamini-Hochberg correction.

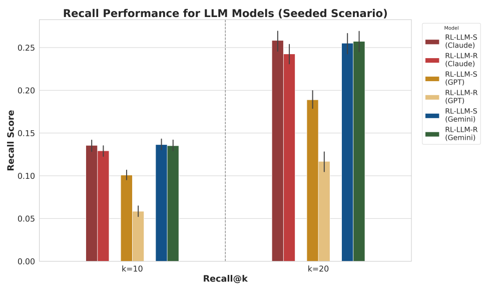
	Claude		**GPT**		**Gemini**	
Metric	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R
precision@10	**0.2712**	0.2585	**0.2018**‡	0.1170	**0.2730**	0.2704
recall@10	**0.1356**	0.1292	**0.1009**‡	0.0585	**0.1365**	0.1352
ndcg@10	**0.2706**	0.2575	**0.2029**‡	0.1099	**0.2727**	0.2714
precision@20	**0.2584**‡	0.2425	**0.1889**‡	0.1168	0.2551	**0.2573**
recall@20	**0.2584**‡	0.2425	**0.1889**‡	0.1168	0.2551	**0.2573**
ndcg@20	**0.2624**‡	0.2473	**0.1938**‡	0.1127	0.2610	**0.2626**

‡Significantly different from the score in the other column for the same LLM. For clarity, inter-LLM significance (e.g., Claude vs. GPT) is described in the text.

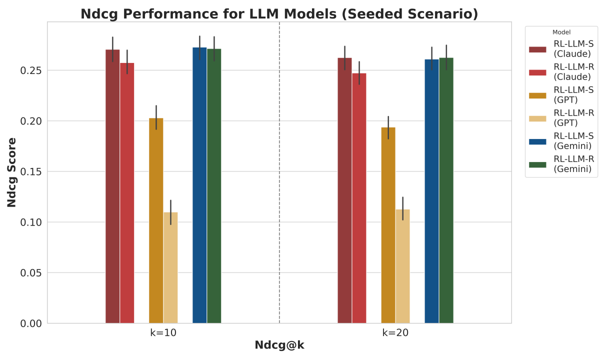
Consistent with the seedless scenario, the agents trained using reward functions from the Gemini model showed robust performance irrespective of the input processing approach. At k=10, the precision scores for the summary-based (RL-LLM-S: 0.2730) and raw-transcript (RL-LLM-R: 0.2704) reward agents were nearly identical, and no performance differences between them were found to be statistically significant for any metric. This suggests the Gemini model's ability to generate effective rewards is not dependent on the input format.



(a) Precision



(b) Recall



(c) NDCG

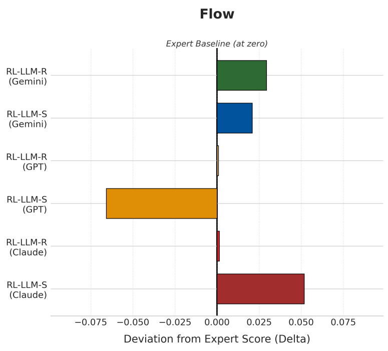
Figure 6.2: Model Performance on Recommendation Quality Metrics in the Seeded Scenario, comparing RL-LLM-S and RL-LLM-R for different LLMs (Claude, GPT, Gemini).

Regarding the choice of LLM, the performance hierarchy established in the seedless scenario remains intact and is statistically robust. The Claude and Gemini reward-based RL models exhibit highly comparable results and are not statistically different from each other on key metrics like nDCG@10. Both models, however, significantly outperform the GPT reward-based models in all configurations. For instance, the nDCG@10 scores for RL-LLM-S (Claude) (0.2706) and RL-LLM-S (Gemini) (0.2727) were significantly higher than for RL-LLM-S (GPT) (0.2029), with p_{adj} < 0.01. This consistent result underscores the performance advantages of the Claude and Gemini reward-based models for this task, both with and without a seed track.

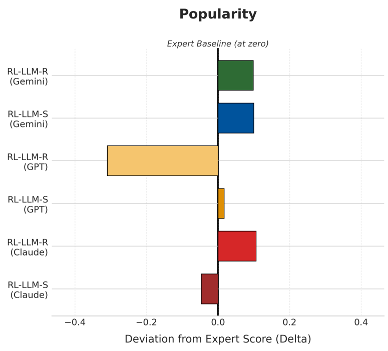
Reward Component Metrics

This section analyzes the alignment of the generated playlists with the expert curatorial profile defined in Section 5.3. The primary measure is the delta score, which represents the absolute difference from the expert profile values. As established in the evaluation design, the goal is not to maximize these component scores, but to match the expert profile as closely as possible. A lower delta therefore indicates a closer alignment with the expert's curatorial style.

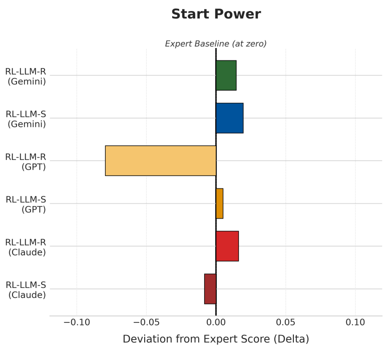
Seedless Playlist Generation (N=0) Table 6.3 along with Figure 6.3 shows the absolute delta scores for Flow, Popularity, Start Power, Diversity, and Novelty, lower values indicate closer alignment to the expert profile. Overall, the RL-LLM-R pipeline yielded smaller deltas on transition, and discovery-focused metrics (Flow, Novelty), while RL-LLM-S achieved tighter matches on engagement and balance oriented metrics (Popularity, Start Power, Diversity). This pattern shows that raw transcripts better preserved nuanced curatorial signals about how songs transition within a playlist, whereas summaries emphasized the expert's explicit priorities for popular, engaging, and varied playlists.



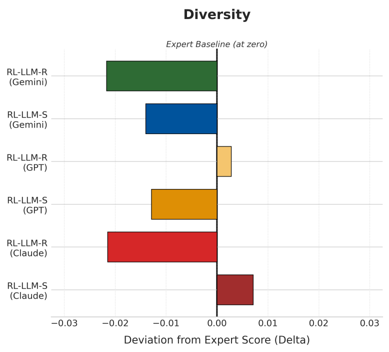
(a) Flow



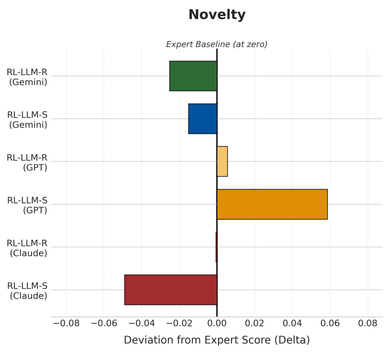
(b) Popularity



(c) Start Power



(d) Diversity



(e) Novelty

Figure 6.3: Deviation of Reward Component Scores for RQ1 in the Seedless Scenario. The plots compare the alignment of each LLM and pipeline combination against the Expert Baseline (zero line).

With Claude, RL-LLM-R produced deltas of 0.0014 for Flow and 0.0006 for Novelty, significantly lower than RL-LLM-S (0.0518 and 0.0491, respectively), indicating that raw transcripts aligned with the expert's song-to-song transition style and their proportion of unpopular songs ("hidden gems"). However, RL-LLM-S matched the expert-profile on Popularity (0.0469 vs. 0.1064), Start Power (0.0084 vs. 0.0161), and Diversity (0.0071 vs. 0.0215) more closely, showing that summaries highlighted the expert's emphasis on familiar, energetic openers and a balanced variety of artists or genres.

For GPT, RL-LLM-R again aligned more on Flow (0.0008 vs. 0.0658), Diversity (0.0028 vs. 0.0129), and Novelty (0.0056 vs. 0.0586), suggesting raw transcripts enabled the reward function to capture curation traits related to song transitions, track variety. By contrast, RL-LLM-S achieved much lower deltas on Popularity (0.0168 vs. 0.3096) and Start Power (0.0050 vs. 0.0795), reflecting that the summaries proved more effective in capturing the expert's intent to favor well-known tracks and strong opening energy.

Table 6.3: Delta Analysis for RQ1 (Seedless): Absolute Difference from Expert Profile. A lower value indicates that the score achieved by the model is closer to the scores computed from the expert curated sequences. The bold marked represented the better scoring approach for each LLM model. The scores marked with * represents the lowest (best) score across all models

	Claude		**GPT**		**Gemini**	
Metric	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R	RL-LLM-S	RL-LLM-R
Flow	0.0518	**0.0014**	0.0658	**0.0008***	**0.0209**	0.0294
Popularity	**0.0469**	0.1064	**0.0168***	0.3096	0.0997	**0.0985**
Start Power	**0.0084**	0.0161	**0.0050***	0.0795	0.0194	**0.0144**
Diversity	**0.0071**	0.0215	0.0129	**0.0028***	**0.0140**	0.0217
Novelty	0.0491	**0.0006***	0.0586	**0.0056**	**0.0151**	0.0252

With Gemini, RL-LLM-S outperformed RL-LLM-R on Flow (0.0209 vs. 0.0294), Diversity (0.0140 vs. 0.0217), and Novelty (0.0151 vs. 0.0252), indicating that summaries conveyed these curatorial aspects more reliably. However, RL-LLM-R held slight advantages on Popularity (0.0985 vs. 0.0997) and Start Power (0.0144 vs. 0.0194), indicating the raw transcripts were able to preserve some expert cues about track familiarity and opening impact.

Overall, these results demonstrate that input format systematically shifted which curatorial components the agents prioritized: raw interviews better captured transition and novelty signals, while summaries emphasized track popularity, initial energy, and overall variety.

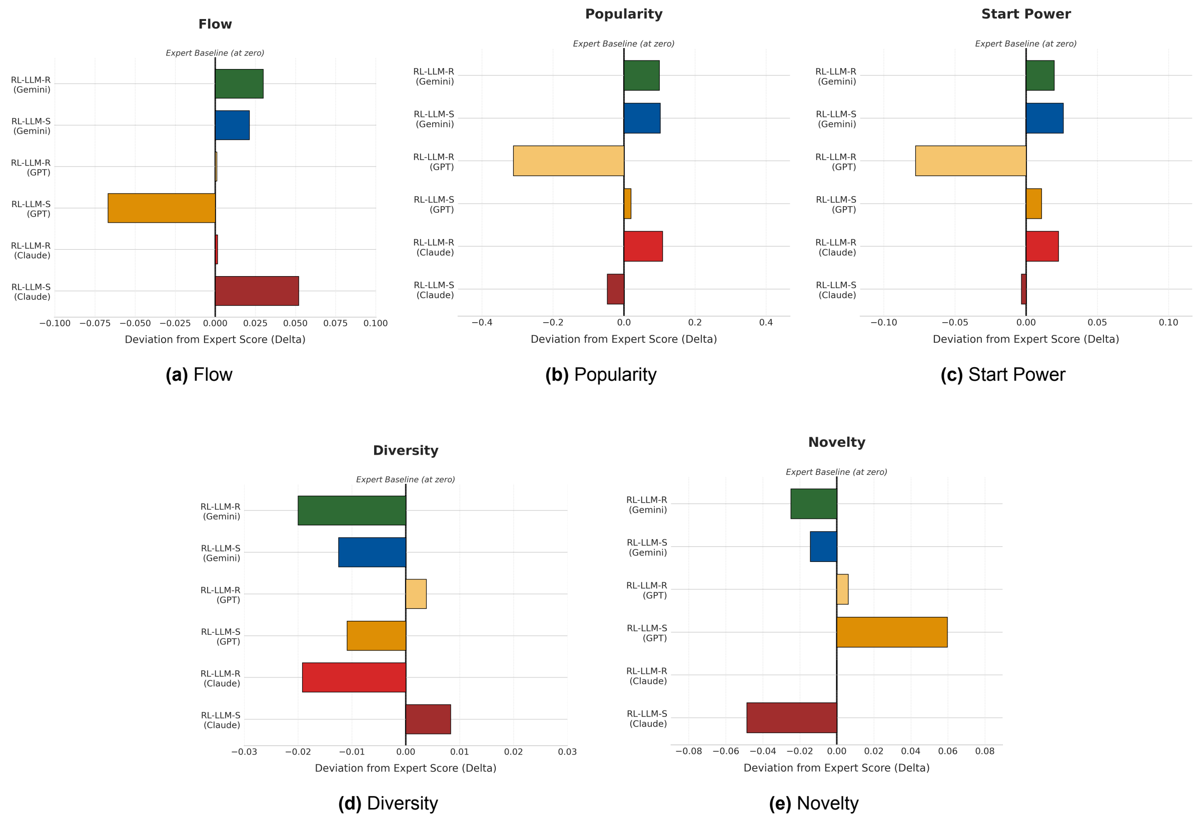
Seeded Playlist Generation (N=1) In the seeded playlist setting, delta scores as shown in Table 6.4 reveal similar performance trends to the seedless case. The RL-LLM-R pipeline continued to yield models with stronger alignment on transition-based and novelty-focused metrics, while the RL-LLM-S models were generally more consistent with the expert profile on popularity, diversity, and start power. This consistency suggests that the choice pipelines influenced the types of curatorial behaviors the agent learns, even when a seed track is provided.

Claude again showed clear differences in curatorial alignment based on input pipeline. The RL-LLM-R model achieved a perfect delta of 0.0000 on Novelty and a very low Flow delta of 0.0014, indicating extremely close adherence to the expert's behavior in both choosing "hidden gems" and maintaining expert-based smooth track transitions. In contrast, RL-LLM-S yielded smaller deltas on Popularity (0.0471 vs. 0.1087), Start Power (0.0034 vs. 0.0226), and Diversity (0.0083 vs. 0.0192), suggesting that the summarized input guided the model to produce playlists that were more popular, opened more strongly, and contained a slightly better mix of artists or genres.

Table 6.4: Reward Component Analysis for RQ1: Seeded Scenario

		Claude		GPT		Gemini	
Metric	**Expert Profile**	**RL-LLM-S**	**RL-LLM-R**	**RL-LLM-S**	**RL-LLM-R**	**RL-LLM-S**	**RL-LLM-R**
Flow	0.8333	0.8853	0.8347	0.7664	0.8342	0.8545	0.8632
Popularity	1.4188	1.3717	1.5275	1.4383	1.1072	1.5205	1.5181
Start Power	0.1128	0.1094	0.1354	0.1236	0.0351	0.1389	0.1325
Diversity	0.7325	0.7408	0.7133	0.7216	0.7363	0.7200	0.7125
Novelty	0.1768	0.1282	0.1768	0.2365	0.1829	0.1624	0.1519

Figure 6.4 provides a visual representation of the deviation in scores form the expert baseline. GPT's reward alignment provided a different result compared to the recommendation quality metrics. RL-LLM-S model showed the largest deviation from the expert profile on almost all metrics, even though it scored significantly higher than RL-LLM-R model on the recommendtion metrics. This could indicate that while the RL-LLM-R model did not pick the same song as the experts, the characteristics of the overall playlist generated was similar to that of the expert. On Flow, the RL-LLM-R model once again produced a near-zero delta (0.0009), far outperforming the RL-LLM-S version (0.0669). The Novelty delta also dropped significantly from 0.0597 (RL-LLM-S) to 0.0061 (RL-LLM-R), highlighting the raw input's strength in reinforcing the expert's discovery-oriented curation. However, this came at the cost of large mismatches on Popularity (delta of 0.3116 for RL-LLM-R vs. 0.0195 for RL-LLM-S) and Start Power (0.0777 vs. 0.0108), suggesting that the raw transcripts failed to capture the expert's preference for well-known or popular opening tracks. These results reinforce GPT's difficulty in balancing multiple curation goals when given unstructured input.



(a) Flow **(b)** Popularity **(c)** Start Power

(d) Diversity **(e)** Novelty

Figure 6.4: Deviation of Reward Component Scores for RQ1 in the Seeded Scenario. Each subplot compares the alignment of the different LLM-guided models against the Expert Baseline.

For Gemini, the differences between pipelines were smaller but still notable. The RL-LLM-S model yielded better scores on Flow (0.0212 vs. 0.0299), Diversity (0.0125 vs. 0.0200), and Novelty (0.0144 vs. 0.0249), indicating better alignment with the expert's stylistic flow, artist variety, and selection of lesser-known songs. Conversely, RL-LLM-R showed slightly lower deltas on Popularity (0.0993 vs. 0.1017) and Start Power (0.0197 vs. 0.0261), but these margins were narrow. Similar to the seedless scenario, Gemini remained relatively unaffected by input format, with both pipelines producing comparable reward alignments across most components.

In the seeded scenario, as well, the pipeline structure influenced the types of curatorial behaviors captured by the models. RL-LLM-R pipelines favored smoothness and novelty, while RL-LLM-S pipelines produced rewards more aligned with popularity, variety, and playlist structure. These effects were most pronounced for GPT and Claude, while Gemini continued to show stable performance across both pipelines.

6.1.2. Qualitative Analysis of LLM-Generated Reward Functions

This section provides a qualitative evaluation of the reward functions generated by our different input pipelines and LLMs. By comparing the identified reward factors and their assigned weights, this analysis links the understanding capabilities of the models to the observed characteristics of their generated playlists. The analysis reveals that the choice of input format, summarized versus raw transcripts, directly influenced the granularity and consistency of the resulting reward functions.

Common Reward Factors Identified

Across a majority of LLMs and both input pipelines, a consistent set of core curatorial principles was identified, suggesting these factors are fundamental to the task. The reward components generated from the summarized transcripts (Pipeline A) and raw transcripts (Pipeline B) are presented in Table 6.5 and Table 6.6, respectively. The most frequently identified factors were:

- **Flow:** This factor was the most consistently identified component and typically received the highest weight (between 0.25 and 0.40). This was in line with our manual observations from the expert interviews. Flow was mainly used by the experts to describe the sequencing of the tracks in the playlist and a factor to consider when determining if a playlist is good. Provided our state representation was mainly based on the Node2vec embeddings, most of the LLM generated reward functions implemented flow as either the average or sum of the cosine similarity between the subsequent songs. The RL-LLM-R approach based LLMs provided a slightly different implementation, taking into account the songs' metadata like 'mood' and 'tempo', which was limited in the number of distinct values present

- **Popularity:** Identified as a primary component with a high weight (often 0.25 to 0.35), this factor reflects the expert principle of balancing curation with audience engagement. Experts confirmed this, stating, *"User engagement would mean that people are familiar with what they listen to. So choosing popular songs...is just easy to listen to…"*. This factor, appearing the sum of the playlist songs' popularity scores or a position-weighted song popularity calculation, where more weightage was given to the popularity of the songs added initially.

- **Artist Diversity:** The models consistently included a factor to avoid artist repetition, assigning it a weight between 0.15 and 0.20. This corresponds directly to expert heuristics such as, *"Apart from the flow, artist separation is a big deal for us."*, or as mentioned by another expert, *"In general for a playlist, I think it's important not to have too much repetitiveness for certain artists"*. It reflects the experts' aim of preventing listener fatigue by not playing the same artist too frequently.

- **Start Strong:** Also identified as "Start Power", many models identified a specific factor for placing popular tracks at the beginning of a playlist, reflecting the expert tactic of engaging the listener early. The weights assigned were generally between 0.15 and 0.20. This aligns with the expert inputs like, *"..maybe you can force order the bigger artist first or the bigger songs first"*.

  The quantitative formula and code provided for computing the start strong reward was consistent across the models. *"..maybe you can force order the bigger artist first or the bigger songs first, and then let the people slowly dive into more discover songs"*

- **Novelty:** Another common factor recognized by most models through Pipeline A and Pipeline B was the need for introducing less popular tracks. The reflects the experts' strategy to surprise the listeners with "hidden gems" and encouraging them to explore new types of songs. In a majority of the generated reward functions this was calculated as the inverse of song popularity.

Reward Functions from Summarized Transcripts (RL-LLM-S)

When prompted with the pre-processed, summarized text from Pipeline A, the LLMs generated reward functions that were highly consistent and focused on a core set of abstract curatorial principles. As shown in Table 6.5, the same fundamental factors emerged repeatedly across all models and runs: Flow, Popularity, Start Strong, Artist Diversity, and Novelty. For example, "Flow" consistently received a high weight (typically 0.25-0.40), aligning with expert feedback that emphasized cohesive transitions. Similarly, "Popularity" was assigned a significant weight (0.25-0.35), reflecting the need to balance curation with audience engagement.

This stability suggests that the summarization process distills the expert interviews into a set of generic agreed-upon rules amongst the experts, filtering out contradictory details. The result is a stable, generalist curation policy built on the most essential principles of playlist creation. Table 6.5 provides insights into the factors identified by the LLMs through the summarized transcripts.

Table 6.5: Reward factors and weights generated by different LLMs from summarized interview transcripts. The table shows five independent generation runs (v1-v5) for each model. The consistency in factors like 'Flow' and 'Popularity' highlights the models' ability to extract core, high-level principles from abstracted text.

Factor	**GPT-o4**					**Gemini 2.5 Pro**					**Claude Opus4**				
	v1	**v2**	**v3**	**v4**	**v5**	**v1**	**v2**	**v3**	**v4**	**v5**	**v1**	**v2**	**v3**	**v4**	**v5**
Flow	0.35	0.30	0.30	0.30	0.25	0.30	0.30	0.30	0.40	0.40	0.30	0.30	0.35	0.35	0.30
Popularity	0.35	0.30	0.30	0.25	0.25	0.30	0.25	0.30	0.35	0.35	0.25	0.25	0.25	0.25	0.25
Start Strong	-	0.20	-	0.15	0.15	0.15	0.20	0.20	0.15	0.10	0.10	0.15	0.15	0.15	0.10
Artist Diversity	0.15	0.20	0.15	0.15	0.20	0.15	0.15	0.15	0.05	-	0.20	0.15	0.15	0.10	0.10
Novelty	0.15	-	0.10	0.15	0.15	0.10	0.10	0.05	0.10	0.15	0.10	0.10	-	0.05	0.05
Artist Popularity	-	-	-	-	-	-	-	-	-	-	0.05	0.05	-	-	0.10
Energy Progression	-	-	-	-	-	-	-	-	-	-	-	-	0.10	0.10	0.05
Genre-variety bonus	-	-	0.15	-	-	-	-	-	-	-	-	-	-	-	-
Discovery bonus	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Reward Functions from Raw Transcripts (RL-LLM-R)

In contrast, when provided with the complete raw transcripts from Pipeline B, the LLMs produced a more diverse and granular set of reward functions. The models identified more specific, conditional heuristics that were absent in the summarized outputs. A notable example is the introduction of Song popularity as a position-weighted score, a rule that reflects the expert tactic of placing popular songs at strategic points in the playlist, rather than just including them anywhere or just at the beginning.

As seen in Table 6.6, the reward structures are also more diverse across runs. For instance, GPT-4o identifies a "Discovery bonus" in four of five runs, while the other LLMs do not. This suggests that the raw text provides a larger and more complex solution space, resulting in more varied interpretations of expert strategy.

This presents a trade-off: summarized transcripts yield consistent reward functions based on core principles, while raw transcripts allow for the extraction of more specific and creative heuristics at the cost of consistency. This finding has significant implications for designing the reward function using LLMs for the RL agent, as the former might create a reliable generalist, while the latter could create a more specialized agent that can mimic individual expert tactics.

Table 6.6: Reward factors identified by LLMs along with their weights with Raw Interview Transcripts

Factor	**GPT-o4**					**Gemini 2.5 Pro**					**Claude Opus4**				
	v1	**v2**	**v3**	**v4**	**v5**	**v1**	**v2**	**v3**	**v4**	**v5**	**v1**	**v2**	**v3**	**v4**	**v5**
Flow	0.25	0.30	0.30	0.30	0.30	0.20	0.25	0.30	0.25	0.25	0.25	0.25	0.35	0.25	0.25
Popularity	-	-	-	-	-	0.25	-	-	0.35	0.30	-	-	0.30	-	-
Start Strong	-	0.05	0.10	-	0.15	0.25	0.10	0.15	0.15	0.20	0.15	0.15	0.10	-	0.15
Song position	0.20	-	-	-	-	-	-	-	-	-	-	-	-	0.15	-
Artist Diversity	0.15	0.20	0.20	0.15	0.15	0.10	0.10	0.15	0.15	-	0.15	0.15	0.20	0.20	0.10
Novelty	-	-	-	-	0.15	0.10	0.05	-	0.10	0.20	0.10	0.10	0.05	0.10	0.10
Song_popularity (position-weighted)	0.25	0.30	0.25	0.35	0.25	-	0.20	0.30	-	-	0.20	0.30	-	0.25	0.20
Artist Popularity	-	-	-	-	-	-	0.15	-	-	-	0.15	0.05	-	-	0.15
Energy Progression	-	-	-	0.10	-	0.10	0.15	-	-	0.05	-	-	-	0.05	0.05
Discovery bonus	0.15	0.15	0.15	0.10	-	-	-	0.10	-	-	-	-	-	-	-

LLM-Specific Behavioral Patterns

While all models identified the core principles, subtle differences in their outputs can be observed:

- **GPT-4o:** This model included a "Discovery bonus" factor when processing raw transcripts, indicating a tendency to identify concepts related to novelty. Its outputs also showed higher structural variety between runs compared to the other models. Appears to be particularly sensitive to concepts of discovery and novelty when processing raw transcripts, frequently including a Discovery bonus factor.
- **Gemini 1.5 Pro:** Tends to produce very stable and robust reward structures. This model tended to produce reward structures with consistently high weights assigned to "Popularity" metrics, suggesting it prioritized engagement-driven heuristics from the text. It consistently assigns a high weight to Popularity metrics, suggesting a strong ability to identify core engagement-driven heuristics from the text.

- **Claude 3 Opus:** This model demonstrated a tendency to identify more relational concepts. It was the only model to consistently extract "Energy Progression" from the summarized text and frequently distinguished between "Artist Popularity" and "Song Popularity".

This presents a clear trade-off: summarized transcripts yield consistent reward factors based on core principles, while raw transcripts result in more specific and creative heuristics at the cost of consistency. This finding has significant implications for designing the RL agent, as the former might create a reliable generalist, while the latter could create a more specialized agent that mimics specific expert tactics.

6.2. RQ2: Alignment of RL Agents with Expert Curation

Having established in Section 6.1 the impact of different LLMs and input processing pipelines on the agent's performance, we now broaden the scope of our investigation. The primary objective of this research is not only to develop a novel method for reward function generation, but also to assess its efficacy relative to existing and simpler approaches. This section, therefore, addresses the second research question: To what extent does RL-LLM agents align with expert curatorial strategies when compared to established baseline methods?

To answer this, we evaluate our two proposed approaches, represented by RL-LLM-S (Gemini) and RL-LLM-R (Gemini) models, against our two baselines: a simple similarity-based approach (N2V-S) and an RL agent with a manually crafted reward function (RL-M). The Gemini LLM was chosen for both approaches as it demonstrated more stable and high-performing results in RQ1. Our findings from Section 6.1 indicated that models specializing in specific curatorial metrics often exhibited diminished overall recommendation quality. In contrast, RL-LLM-S (Gemini) and RL-LLM-S (Gemini) achieved the highest scores on primary recommendation metrics like $ndcg@k$, while also demonstrating comparable performance across the other recommendation quality and reward component metrics.

6.2.1. Quantitative Analysis

Similar to RQ1, the quantitative analysis first examines the performance through the lens of recommendation quality metrics and then into a comparison of expert-curatorial alignment, in the seedless and seeded playlist completion scenarios.

Recommendation Quality Metrics

We first assess the models' ability to replicate expert-curated playlists using standard recommendation accuracy metrics. This evaluation is helps in understanding the practical performance of our proposed models (RL-LLM-S and RL-LLM-R) in comparison to the baselines (N2V-S and RL-M).

Seedless Playlist Generation (N=0): In the seedless scenario, where models generate a 20-song playlist from only a title, the results in Table 6.7 reveal a clear and statistically significant performance hierarchy. A substantial gap exists between the RL models and the Node2Vec baseline. The N2V-S model's precision@10 of 0.0582 indicates it placed, on average, fewer than one expert-selected track for every playlist generated. Its correspondingly low nDCG@10 score of 0.0493 further suggests that its simple similarity-based strategy is insufficient for the cold-start generation task, as it fails to identify relevant tracks and rank them effectively.

The manually-crafted reward-based RL model, RL-M, demonstrates a considerable improvement over N2V-S, achieving a precision@10 of 0.2405, corresponding to roughly 2.4 expert-approved tracks in its top ten. However, it is statistically outperformed by the LLM-guided models. For instance, the RL-LLM-S (Gemini) model achieved a precision@10 of 0.2731, placing approximately 2.7 correct tracks. The higher recall@10 (0.1365 vs. 0.1203) and nDCG@10 (0.2729 vs. 0.2437) scores confirm that the LLM-based reward function not only enabled the agent to cover a larger portion of the expert's choices but also to rank them more effectively.

Table 6.7: Recommendation Quality in the Seedless Scenario (N=0). Scores are annotated with superscript letters (a, b, c) to denote statistical significance based on a Wilcoxon signed-rank test (α=0.05). Models that do not share a common letter are significantly different. The analysis reveals three distinct performance tiers, with our proposed RL-LLM models (a) significantly outperforming both the manual RL-M (b) and N2V-S (c) baselines. Best scores are in **bold**.

Metric	N2V-S	RL-M	RL-LLM-S (Gemini)	RL-LLM-R (Gemini)
ndcg@10	0.0493^c	0.2437^b	**0.2729**^{a}	0.2718^a
precision@10	0.0582^c	0.2405^b	**0.2731**^{a}	0.2709^a
recall@10	0.0291^c	0.1203^b	**0.1365**^{a}	0.1354^a
ndcg@20	0.0623^c	0.2338^b	0.2699^a	**0.2704**^{a}
precision@20	0.0724^c	0.2278^b	0.2687^a	**0.2693**^{a}
recall@20	0.0724^c	0.2278^b	0.2687^a	**0.2693**^{a}

For k=20, N2V-S remains poor, placing less than 1.5 expert-selected tracks on average (precision@20 of 0.0724). RL-M significantly improves this to approximately 4.5 expert-selected tracks (precision@20 of 0.2278). The LLM-guided models, RL-LLM-S (Gemini) and RL-LLM-R (Gemini), perform best, consistently picking around 5.4 expert-selected tracks (precision@20 of 0.2687 and 0.2693 respectively), further demonstrating their ability to generate relevant and well-ranked playlists from scratch.

The statistical significance testing formally validates these observations, identifying three distinct performance tiers (p_{adj} < 0.05). The two Gemini-based RL-LLM models, which are statistically indistinguishable from each other, significantly outperform the RL-M model, which in turn significantly outperforms the N2V-S baseline. This confirms the value of the LLM-generated reward signals over both a manually specified one and a simple embedding-based approach for this task.

Seeded Playlist Generation (N=1) When a seed track is provided, the performance hierarchy shifts , as shown in Table 6.8. The N2V-S baseline becomes the top performing model across all recommendation quality metrics, suggesting the task becomes less about interpreting a broad concept (the title) and more about local, similarity-based continuation. The N2V-S model's precision@10 of 0.2966 indicates it placed nearly three expert-approved tracks in the top ten recommendations.

However, the significance tests presents an additional insight. At the k=10 level, there is no statistically significant difference between the N2V-S model and the two RL-LLM models, placing them in the same performance tier. This suggests that for shorter playlists containing an initial track, the next-track predictions based on the direct signal from embedding similarity is as effective as the more complex RL policies. All three of these models do, however, significantly outperform the RL-M baseline, whose precision@10 of 0.2398 corresponds to only 2.4 correct tracks.

Table 6.8: Recommendation Quality in the Seeded Scenario (N=1). Scores are annotated with superscript letters (a, b, c) to denote statistical significance (α=0.05). Models that do not share a common letter are significantly different. For top-10 metrics, N2V-S and the RL-LLM models (a) are statistically indistinguishable and significantly outperform RL-M (b). For top-20 metrics, N2V-S (a) is the clear top performer. Best scores are in **bold**.

Metric	N2V-S	RL-M	RL-LLM-S (Gemini)	RL-LLM-R (Gemini)
ndcg@10	**0.2992**^{a}	0.2427^b	0.2727^a	0.2714^a
precision@10	**0.2966**^{a}	0.2398^b	0.2730^a	0.2704^a
recall@10	**0.1483**^{a}	0.1199^b	0.1365^a	0.1352^a
ndcg@20	**0.2955**^{a}	0.2266^c	0.2610^b	0.2626^b
precision@20	**0.2927**^{a}	0.2175^c	0.2551^b	0.2573^b
recall@20	**0.2927**^{a}	0.2175^c	0.2551^b	0.2573^b

At the k=20 level, we observe a clear hierarchy. The N2V-S model's advantage over the RL models becomes statistically significant, solidifying its strength in the seeded playlist continuation scenario. Amongst the RL models, models from both the LLM based approaches, significantly outperforming the RL-M model. Interestingly, the performance of the RL agents did not uniformly improve with a seed track; this suggests that the provided seed may at times constrain the learned policy in a way that hinders its ability to match the expert sequence. Nevertheless, the results consistently show that the LLM-guided agents significantly outperform the manually-tuned agent, while the summary-based (RL-LLM-S) and raw-transcript-based (RL-LLM-R) pipelines remain statistically indistinguishable for the Gemini model.

Reward Component Metrics

While we established how the models perform with respect to the song selection, we further analyze the characteristics of the playlists generated by the model with respect to our reward component metrics. By evaluating against the reward component metrics, we can measure how closely each model's output aligns with the complex curatorial profile derived from expert interviews. The primary metric used is the delta score, the absolute difference from the expert profile benchmark, where a lower value indicates a closer alignment. The accompanying deviation plots in Figures 6.5 and 6.6 visualize these differences, with the zero-line representing a perfect match with the expert profile.

Seedless Playlist Generation (N=0) The delta analysis in Table 6.10 provides insight into how each model's output aligns with the target curatorial profile. An analysis of the reward components in the seedless scenario reveals the distinct operational logic and resulting trade-offs of each model. By correlating the delta scores from Table 6.10 with the achieved raw scores in Table 6.9, we can develop a critical understanding of their respective alignment strategies.

The N2V-S baseline, which does not explicitly model the expert strategies, shows divergence from the expert profile on several metrics. Its large delta on Popularity (0.4975) is because its achieved score (0.9213) is substantially lower than the expert target (1.4188). Its Start Power delta (0.1128) is simply the expert target itself, as the model's raw score is 0.0000, reflecting its inability to factor in track position or popularity. Its performance on Flow (delta 0.0519) is moderate, suggesting that while embedding similarity is a reasonable proxy for coherence, the model tends to slightly over-optimize it (raw score 0.8852 vs. target 0.8333).

Table 6.9: Reward Component Analysis: Seedless Scenario. This table shows the raw scores for each model against the expert profile.

Metric	Expert Profile	N2V-S	RL-M	RL-LLM-S	RL-LLM-R
Flow	0.8333	0.8852	0.8110	0.8542	**0.8347**
Popularity	1.4188	0.9213	1.3786	1.5185	1.5173
Start Power	0.1128	0.0000	0.1144	1.322	0.1272
Diversity	0.7325	0.6926	0.7294	0.7185	0.7108
Novelty	0.1768	0.1376	0.1989	0.1617	0.1762

Note: The RL-LLM-S and RL-LLM-R columns show results from their respective Gemini-guided models. Bold values indicate the score closest to the Expert Profile.

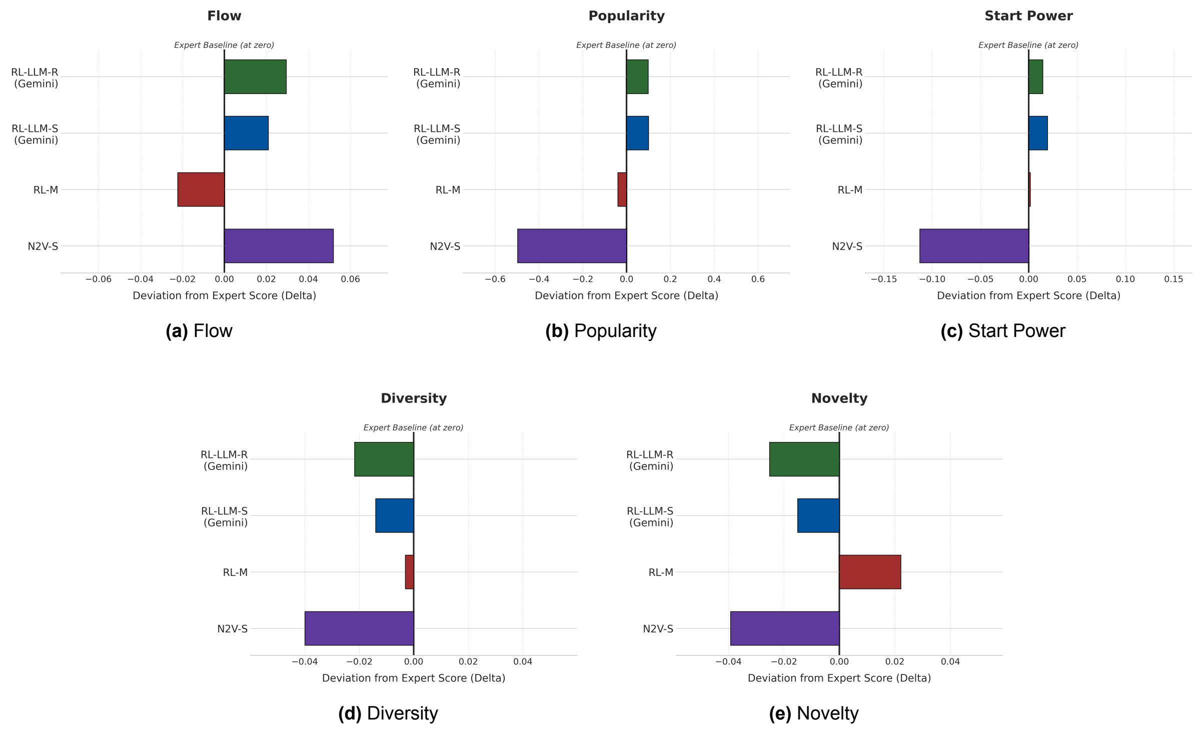
The RL-M model provides a compelling case study in manual reward engineering, with it achieving the lowest deltas on three of the five metrics. Its top performance on Diversity (delta 0.0031) and Start Power (delta 0.0016) stems from its ability to learn and execute clear rules. The Diversity score of 0.7294 is nearly identical to the expert target of 0.7325. The Start Power score of 0.1144 is also remarkably close to the 0.1128 benchmark, a likely emergent property from optimizing for its high-weighted factors like Popularity and Coherence.

It also achieves the best alignment on Popularity (delta 0.0402), with its raw score of 1.3786 being the closest to the expert profile (1.4188). This is counterbalanced by its performance on Novelty, where its score of 0.1989 overshoots the target of 0.1768, resulting in a delta of 0.0221. This illustrates the explicit trade-off dictated by its fixed weights: the agent prioritizes hitting the Popularity target, which comes at the cost of slightly over-representing novel tracks to satisfy its secondary Novelty objective.

Table 6.10: Delta Analysis (Seedless): Absolute Difference from Expert Profile. A lower value indicates that the score achieved by the model is closer to the scores computed from the expert curated sequences and are in **bold**.

Metric	N2V-S	RL-M	RL-LLM-S	RL-LLM-R
Flow	0.0519	0.0223	**0.0209**	0.0294
Popularity	0.4975	**0.0402**	0.0997	0.0985
Start Power	0.1128	**0.0016**	0.0194	0.0144
Diversity	0.0399	**0.0031**	0.0140	0.0217
Novelty	0.0392	0.0221	**0.0151**	0.0252

The LLM-guided models exhibit a different set of trade-offs, showing better performance on abstract metrics. The RL-LLM-S model achieves the best delta for Flow (0.0209) and Novelty (0.0151). Its Novelty score of 0.1617 is very close to the expert target (0.1768), suggesting the LLM better captured the nuanced balance between discovery and popularity described in the interviews. However, this comes at the cost of precision on other metrics. Both RL-LLM-S and RL-LLM-R produce playlists with higher Popularity scores (1.5185 and 1.5173, respectively) than the expert target (1.4188). This leads to larger deltas on this metric (0.0997 and 0.0985) compared to RL-M. This suggests the LLM-generated reward functions may have interpreted “popularity“ as a general principle to be maximized, rather than a specific level to be targeted, resulting in less precise alignment to that specific value.



(a) Flow **(b)** Popularity **(c)** Start Power **(d)** Diversity **(e)** Novelty

Figure 6.5: Deviation of Reward Component Scores from the Expert Profile for the Seedless Scenario. Each subplot shows the performance for a single metric against the Expert Baseline (zero line).

The seedless scenario highlights that no single model achieves superior alignment across all curatorial dimensions. The RL-M model achieved a higher alignment with more quantifiable and structural components of the expert profile, such as popularity levels and adhering to rules for artist diversity. The RL-LLM models offer a different kind of alignment, one that is broader but less numerically exact. This could be an indication that the LLM-based rewards aligned more on subjective and stylistic aspects of the expert profile, such as capturing a musical “flow“ and balancing track discovery with familiarity (Novelty).

Seeded Playlist Generation (N=1) We further evaluate the model generated playlists in the seeded scenario. An analysis of the raw scores in Table 6.11 and the corresponding deltas in Table 6.12 highlights a divergence between the similarity-based N2V-S model and the reward-driven RL models.

Although the N2V-S model achieves high recommendation accuracy in the seeded scenario, this performance is accompanied by substantial deviations from key curatorial benchmarks. A significant deviation is observed in Diversity, where the raw score of 0.1125 results in a large delta of 0.6200. This can be attributed to a "homogenization" effect, where the model's reliance on embedding similarity leads it to repeatedly select tracks from the same artist or album as the seed song provided. Similarly, the model exhibits a lack of control over Start Power, yielding a delta of 0.6083. The raw score of 0.7211 is not a product of a learned strategy but an result of the seed track's own popularity, wherein the N2V-S picks then adds songs with similar popularity. Conversely, its local optimization strategy results in the closest alignment on Popularity (delta 0.0336), where its raw score (1.3852) is nearest to the expert profile (1.4188).

In contrast, the RL-M model maintains high fidelity to the target profile, exhibiting considerable robustness. It achieves the lowest deltas for Start Power (0.0064) and Diversity (0.0011), with its raw scores (0.1192 and 0.7314, respectively) closely matching the expert benchmarks. This illustrates the efficacy of its explicit, rule-based design in mitigating the homogenization and control issues observed in the N2V-S model.

Table 6.11: Reward Component Analysis: Seeded Scenario. This table shows the raw scores for each model against the expert profile.

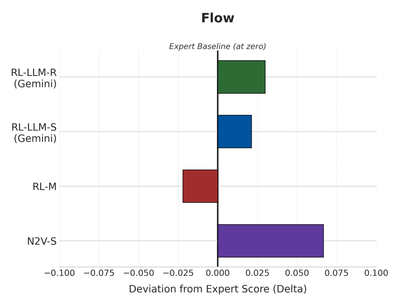
Metric	Expert Profile	N2V-S	RL-M	RL-LLM-S	RL-LLM-R
Flow	0.8333	0.9000	0.8112	**0.8545**	0.8632
Popularity	1.4188	**1.3852**	1.3786	1.5205	1.5181
Start Power	0.1128	0.7211	**0.1192**	0.1389	0.1325
Diversity	0.7325	0.1125	**0.7314**	0.7200	0.7125
Novelty	0.1768	0.1264	0.1994	**0.1624**	0.1519

Note: The RL-LLM-S and RL-LLM-R columns show results from their respective Gemini-guided models. Bold values indicate the score closest to the Expert Profile.

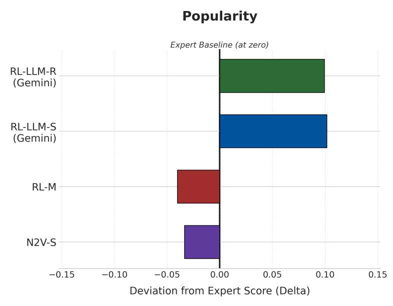
The LLM-guided agents on the other hand, maintained low deltas on structural metrics like Start Power and Diversity. The RL-LLM-S model attains the lowest deltas for Flow (0.0212) and Novelty (0.0144). Its raw Novelty score of 0.1624 is the closest to the expert target of 0.1768, suggesting its tendency to focus on addtional criteria like maintaining artist diversity as well as avoiding popularity bias.

Table 6.12: Delta Analysis (Seeded): Absolute Difference from Expert Profile. A lower value indicates that the score achieved by the model is closer to the scores computed from the expert curated sequences and are in **bold**.

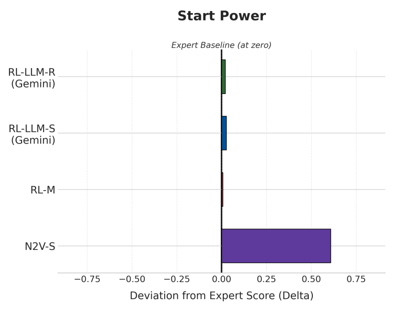
Metric	N2V-S	RL-M	RL-LLM-S	RL-LLM-R
Flow	0.0667	0.0221	**0.0212**	0.0299
Popularity	**0.0336**	0.0402	0.1017	0.0993
Start Power	0.6083	**0.0064**	0.0261	0.0197
Diversity	0.6200	**0.0011**	0.0125	0.0200
Novelty	0.0504	0.0226	**0.0144**	0.0249



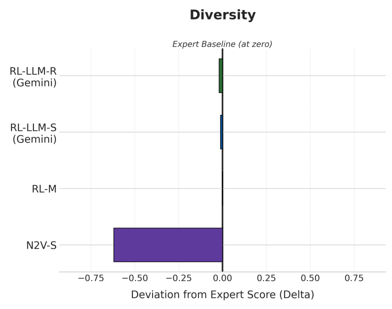
(a) Flow



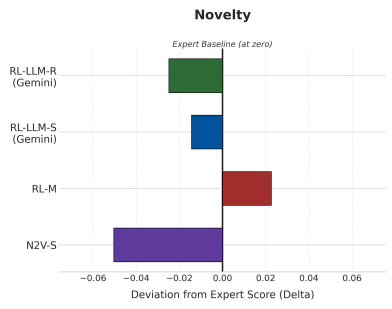
(b) Popularity



(c) Start Power



(d) Diversity



(e) Novelty

Figure 6.6: Deviation of Reward Component Scores from the Expert Profile for the Seeded Scenario. Note the significant deviation of the N2V-S model on Start Power and Diversity.

The seeded scenario clearly differentiates the models based on their strategic scope. The performance of the N2V-S model illustrates that a similarity-driven optimization, can be more effective on the playlist continuation task. By evaluating the playlist as a whole, their reward-driven policies can integrate a local constraint without deviating from fundamental curatorial principles. The RL-M model is particularly effective at preserving the explicit structure of the playlist, while the RL-LLM models show a notable capacity for maintaining its intended style, highlighting the different capabilities of manually-engineered versus interpretively-derived reward systems.This indicates that while choosing songs that are closer in the embedding space can result in more expert-selected tracks to be added to the models' playlists, the tracks that do not overlap could be different from the type of songs and the overall composition of the experts' playlists.

6.2.2. Stability of the RL Agents

To establish the reliability of our findings, each RL agent was trained five times with different random seeds. The learning curves from this process, as illustrated by the stable convergence of the RL-LLM-S (Claude) agent in Figure 6.7, showed minimal variance across runs. This stability, further detailed with the evaluation metric scores obtained by the 5 instances of the agents (presented in Appendix D), confirms the robustness of the models. All results presented above for each RL agent is therefore based on these averaged scores, allowing for a reliable comparison.

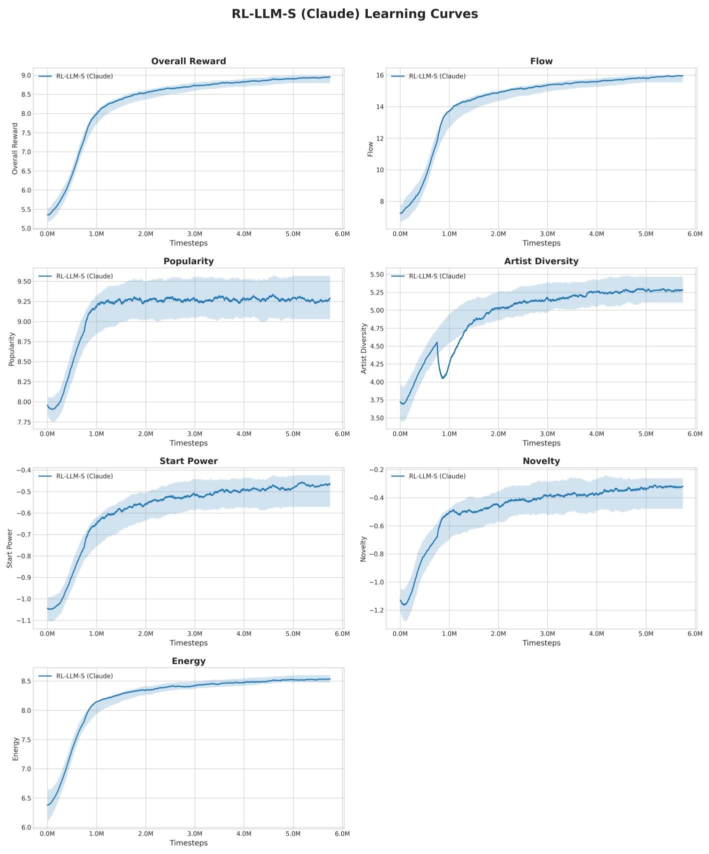


Figure 6.7: Illustrative learning curves for the RL-M agent, showing stable convergence across five independent runs.

7
Discussion

This chapter provides a discussion of the results presented in Chapter 6, interpreting their implications for our research questions and broader applications, within academic research and industry. We explore how our findings answer the central research question and its sub-questions, highlight other implications, and finally, address the limitations of this study.

7.1. Impact of Transcript Processing and LLM Choice

Our investigation into leveraging LLMs for interpreting expert curatorial explanations and generating computable reward components (RQ1) give insights into the capabilities and limitations of the proposed summarization pipeline.

7.1.1. The Summarization Pipeline: Function and Implications

The ”Summarization-First Approach” (RL-LLM-S) was designed to combine and condense multiple, unstructured expert interviews into a concise, pre-verified summary before LLM processing. This pipeline consistently led to the generation of stable and generalized reward functions, centered on the more commonly mentioned and recognized curatorial principles such as “Flow“, “Popularity“, “Artist Diversity“, “Start Strong“ and “Novelty“. The reward factors identified through the summarization approach across various LLMs remained consistent, both within multiple runs of the LLM (v1-v5, Table 6.5 and across different LLM instances. This suggests that the summarization process narrowed down the LLMs’ focus on the concepts that represented a common consensus across the experts, thus leading to a more ”generalist” curation policy.

Quantitative Impact on Recommendation Quality: One of the questions we try to answer is whether inclusion of the summarization step offered an advantage. The results were mixed across the LLMs, suggesting that the choice of LLMs also influenced the effect of this approach. For GPT-based Agents, the summarization pipeline demonstrated a statistically significant advantage in the seedless scenario. This indicates that for GPT models, providing a pre-summarized, less noisy input resulted in a reward signal that guided the RL agent to learn a more effective expert policy, validating the benefit of the summarization step for this specific LLM. The Claude-guided agents showed an improvement in recommendation accuracy on longer playlists when moving from raw to summarized inputs. Gemini-guided Agents’ robustness and stability across both input pipelines, with minimal statistical differences in performance, brings out an interesting insight. The similarity in performance across both pipelines could suggest that Gemini possesses strong capabilities in processing diverse textual inputs effectively, or, as a critical observation, that its internal processing might inherently perform a similar summarization step as Pipeline A. Given that Gemini was used also for the summarization stage of the RL-LLM-S pipeline, this raises the question of whether an external summarization step offers an added advantage when using Gemini, implying its internal mechanisms already perform a similar summarization to retrieve relevant information from qualitative data. This finding necessitates investigating whether LLMs internally perform a summarization step when processing raw transcripts to better understand their mechanisms for knowledge formalization.

7.1.2. LLM Information Processing: Consistency or Granularity?

While, the reward factors generated from the raw transcripts, overlapped quite a bit with the one generated from the summarization pipeline, it did capture some additional set of heuristics. This was evident in the emergence of highly specific factors like “Song popularity as a position-weighted score“, “Discovery bonus“, and “Genre-variety bonus“. Additonally, we observed that resulting reward factors from RL-LLM-R approach were less consistent across runs, indicating that the LLMs focused on different specific details in each iteration. This suggests that while LLMs possess the capability to capture nuanced information, this comes at the cost of less predictable outputs when the input is unconstrained and highly variable.

7.1.3. Bridging Tacit Knowledge to Computable Code

Beyond reward function generation, the analysis of the generated reward code revealed that, given sufficient context about the agent’s state, action space, and available data, LLMs were capable of generating functional code with minimal errors. The observed errors, such as references to non-existing data or incorrect variable references, were largely addressable and could be resolved through the iterative code refinement step. Additionally, providing the LLMs with a skeletal structure for the reward code resulted in more consistent and easier to implement code, though potentially at the cost of excluding exhaustive or highly creative approaches (e.g., using expert-criteria mappings for playlist titles). The findings from these results suggest that LLMs are capable of capturing expert-based curatorial criteria, though their ability to translate abstract and tacit concepts into a structured, executable format requires further investigation, given the lack of consensus on how to quantify these metrics.

7.2. RQ2: Alignment of RL Agents with Expert Curation

Our second research question evaluates the practical efficacy of our proposed RL-LLM models by comparing them against two baselines: a non-RL, similarity-based model (N2V-S) and an RL agent guided by a manually-engineered reward function (RL-M). The results reveal that the effectiveness of each approach is highly dependent on the playlist generation context, specifically whether the playlist generation task is seedless (cold-start) or seeded (continuation).

7.2.1. Performance of Model in Seedless Playlist Generation

In the seedless generation scenario, which tests the models’ ability to generate playlists from scratch using a thematic title, the results revealed a clear and statistically significant performance hierarchy. The data presented in Table 6.7 inidicates that the RL-LLM models outperformed the baselines models on the recommendation metrics.

This outcome implies that for generating theme-based playlists a simple similarity-based approach is insufficient. The RL agents, which leverage the same song embeddings but are guided by a long-term reward signal, learn a more effective policy. The success of the LLM-guided agents over the manually-tuned one further suggests that the LLM-derived reward function captures the experts’ preferences more effectively than the hand-crafted rules of the RL-M agent. However, while the improvement is statistically significant, the difference in raw performance scores shows the practical gain as modest, translating to less than one additional expert-approved track in a 20-song playlist. This raises questions about the practical magnitude of the advantage. While our findings validate the hypothesis that LLM-derived rewards can enhance performance in the cold-start problem, the degree of this improvement suggests that a well-designed manual reward function can still be highly competitive, and the additional complexity of the LLM pipeline may yield only marginal gains in this context.

Moreover, this comparison is complicated by the reward component analysis, which does not mirror the same performance hierarchy. The manually-tuned RL-M agent, despite its lower recommendation accuracy, achieved a closer alignment to the expert profile on quantifiable metrics like Popularity and Diversity. This suggests the RL-M agent was better at optimizing for the explicit, rule-based definitions we provided, even if the tracks specifically differed. In contrast, the RL-LLM models’ superiority in overall accuracy may indicate they learned a policy that better reflects the experts’ true intent, even if that intent was not perfectly captured by our specific, and simplistic, quantitative formulas for abstract concepts like “flow“ or “novelty“.

7.2.2. Performance of Model in Playlist Continuation

When the task shifted from playlist generation to seeded playlist continuation, the performance hierarchy changed. The non-RL N2V-S baseline outperformed the RL-based models on recommendation quality metrics, achieving an nDCG@20 of 0.2955, significantly outperforming the RL-LLM (0.2626) and RL-M (0.2266) models .

This result highlights a fundamental distinction between local, greedy optimization and global, policy-based optimization. The N2V-S model, when provided with a seed track, shows a substantial improvement in its recommendation accuracy and ranking. This indicates that a simple similarity based heuristic, can be more effective at predicting the next likely tracks in a sequence. On the other hand, the presence of a seed track did not improve the accuracy of any of the RL agents. This could be a result of the agents learned policy, which takes into consideration multiple factors from the reward, and not just the information gained from the seed song. While the N2V does have a statistically significant improvement over the RL agents, a comparison of the raw scores reveal that the almost same number of correct songs are picked by both approaches in the seeded scenario (5,9 vs 5,2 out of 20 tracks).

However, this picture of the N2V-S model's superiority is reversed when examining the reward component scores. The N2V-S model, despite its high accuracy, showed more deviation from the expert profile on the reward component scores. Its greedy strategy led to playlist homogenization, with low artist diversity scores, revealing that the overall characteristic of its playlists did not align with experts on our simple metrics. The RL agents, in contrast, demonstrated closer characteristic alignment with these global curatorial principles.

7.3. Industrial and Practical Applications

The motivation for this research was rooted in the industrial challenge of scaling expert curation at XITE, where a small team of music experts is tasked with creating a large volume of high-quality playlists to meet increasing user demand. The findings suggest that the proposed RL-LLM framework should not be viewed as a replacement for human curators, but rather as a potentially valuable assistive tool. The interviews as well as the reward component metric scores of the models revealed that the expert curatorial process is deeply abstract, relying on tacit knowledge, and intuitive judgments that are difficult to fully formalize into quantitative reward signals. The models learn from a static snapshot of this knowledge, whereas human experts continually adapt to new trends and cultural shifts. Therefore, a realistic application of this system would be to augment the expert workflow, for instance, by generating a "first draft" of a playlist which the experts can further refine. This has the potential to reduce the time required for initial track selection, allowing experts to focus on more creative and strategic tasks, especially during periods of high demand.

However, translating these research findings into a production environment requires an assessment of the operational costs against the potential benefits. The adoption of the RL framework is fundamentally constrained by its high computational cost. The expenditure required to train an RL agent, in our case, 5 million timesteps for each configuration, is orders of magnitude greater than that of the one-time training for the simpler Node2Vec model. Deploying and maintaining such a system requires a substantial investment in computational resources. This significant cost necessitates further evaluation of whether the performance gains justify the expenditure.

7.4. Limitations of our work

While this study provides a novel framework, its limitations must be acknowledged as they could serve as directions for future research. A main limitation lies in the study's data and scope, which impacts the generalizability of the findings. The reliance on an internal dataset from XITE meant that metrics like song popularity had to be proxied, creating a potential disconnect from the real-world signals (e.g., Billboard charts, Spotify data) that experts use. Furthermore, the expert knowledge was elicited from a small, homogeneous group of eight curators from a single organization. Their curatorial strategies, while valuable, may not be universally representative, and the framework's effectiveness should be validated with a more diverse expert pool and across different music platforms.

A second limitation was the metrics used to compare the playlists' characteristics. Our operationalization of concepts like "flow" and "novelty" resulted in overly simple definitions of these abstract concepts. For instance, when using cosine similarity as a proxy for "flow", a comparison between the two playlists gets reduced to their average track similarity, a behavior that is not always representative of an expert's strategy. An alternate quantitative representation of these metrics might lead to different conclusions about playlist alignment. This necessitates the need for more human-based evaluation. A further limitation is the foundational assumption of our work, that experts' strategies in interviews perfectly mirror their actual behavior. There can be a significant gap between an expert's stated process and their actions when creating a playlist. Future work could address this by augmenting interviews with expert demonstrations (i.e., analyzing playlists curated by those same experts), which would ground the LLM's understanding in actual expert behaviour.

Finally, the study was confined to assessing playlists of a fixed 20-song length, which may not be enough to capture the variation of the expert playlists, which can range up to over 4000 tracks in length. This also indicates that our evaluation may not fully assess the theoretical advantage of RL, which is its ability to optimize for cumulative rewards over long durations. Furthermore, the training and evaluation were performed on a narrow set of themes defined by explicit metadata like decade and genre. Future research should therefore extend the evaluation to playlists of greater and more variable lengths and broaden the training scope to include more abstract, subgenre, or mood-based titles. This would provide a more robust assessment of the learned policy's generalizability and long-term performance.

8 Conclusion and Future Work

This thesis proposed a novel methodology for integrating the tacit knowledge of expert curators into a RL framework for theme-based playlist generation. The primary objective was to bridge the gap between the nuanced, qualitative strategies of human experts and the quantitative reward functions required by RL agents. The research was structured around two main quesions. Firstly, it explored how LLMs could be leveraged to interpret natural language explanations from curators and generate effective reward functions, comparing a pipeline that used summarized interview data against one that used raw transcripts. Secondly, it assessed the extent to which an RL agent guided by these expert-informed rewards could generate playlists that align with professional curation standards in comparison to baseline models.

The investigation into using LLMs for reward generation revealed a trade-off between the two processing pipelines. The summarization-first pipeline, which processed a condensed and verified summary of expert interviews, consistently produced stable and generalist reward functions focused on high-level, consensual principles like flow, popularity, and diversity. In contrast, the direct-from-raw pipeline, which provided the LLM with the complete, unstructured transcripts, allowed for the extraction of more varied, and specific factors like Genre-variety bonus. This came at the cost of consistency, indicating that while raw text can be more descriptive, it can also lead to less predictable reward structures. The choice of LLM also proved to be a significant factor; models from the Claude and Gemini families consistently outperformed the GPT-based models used in this study.

The results obtained from the comparative evaluation of the playlist generation models varied based on the task. In the seedless playlist generation, the proposed RL-LLM models significantly outperformed both the manually-tuned RL agent (RL-M) and a similarity-based baselines (N2V-S) on standard recommendation quality metrics. This suggests our agent, trained on expert insights, was able to create better expert-aligned playlists. The LLM-derived reward functions proved more effective at capturing the multi-objective nature of expert curation than a hand-crafted reward formula. However, in the seeded playlist continuation task, the performance hierarchy inverted. The simple, similarity-based N2V-S baseline achieved the highest scores on recommendation accuracy metrics, suggesting that for local, next-track prediction, a greedy similarity heuristic can be more effective when an initial context is provided.

The N2V-S model, however, showed misalignment with the expert curatorial profile on different heuristics, thus producing playlists with extremely low artist diversity. The RL-based agents, conversely, demonstrated better alignment with curatorial principles such as diversity and balancing popularity with novelty, even in the seeded scenario.

The primary contributions of this thesis are threefold. It introduces a methodology for translating tacit expert knowledge into executable reward function code using LLMs. Furthermore, it provides a comparative analysis of knowledge processing pipelines, revealing a trade-off between the consistency of rewards from summarized text and the granularity in rewards from raw text. Third, it offers an empirical comparison of different foundational LLMs for the task of converting tacit knowledge to reward function for RL. By establishing a baseline methodology, this research lays the groundwork for subsequent research to integrate experts' tacit knowledge in various domains.

The study does have some limitations. The use of proxied data for measuring metrics like song popularity could have influenced our results. Furthermore, the operationalization of abstract concepts like "flow" into simple quantitative metrics for evaluation is a simplification of complex human perception. This brings forth the need for more concrete experiments and metrics to draw more definitive conclusions.

8.1. Future Work

The findings and limitations of this study suggest several directions for future research. These avenues focus on validating the approach with real users, enhancing the system's semantic understanding, incorporating expert demonstrations, exploring our methodology's broader applications.

8.1.1. Online Evaluation and Human-in-the-Loop Refinement:

One of the most important next steps would be to evaluate our models in a online setting. While necessary for initial validation, an offline setting cannot fully capture the dynamic nature of user satisfaction. Conducting an online A/B testing, similar to the setting proposed in [109], would help measure how the playlists generated by the RL-LLM framework compare to the expert-curated playlists. This would provide more conclusive results on our models' ability to generate high-quality playlists. Tracking key user engagement metrics, such as session length, skip rates, and user feedback, can help determine if the observed offline improvements translate into a genuinely better user experience.

Furthermore, the current reward generation process is static. A more advanced implementation could incorporate a human-in-the-loop step our methodology, as demonstrated in [26]. This would involve iterative reward refinement, where experts provide natural language feedback on generated playlists ("the energy drops too quickly here") to dynamically update the reward function.

8.1.2. Improving Semantic Understanding and Model Capabilities

The system's performance is dependent on its ability to understand the semantic content of both the songs and the playlist themes. The current Node2Vec embeddings, while effective, does not capture meanings from song lyrics or the playlist titles. Incorporating song lyrics, through RNN-based feature extraction mechanisms, as proposed in [110], could help the agent gain more information about the songs. Similarly, our current approach is limited for themes defined by explicit song metadata (e.g., "90s Pop"). To handle more abstract themes like "workout motivation" or "happy hits", the system requires a deeper understanding of the semantic meaning of playlist titles. Building on Yürekli et al.'s work [54] which utilizes semantic meaning from playlist titles, future research could integrate semantic representations of the titles into the agent's state.

8.1.3. Incorporating Expert Demonstrations

As noted earlier, a main assumption of our paper is the alignment between expert descriptions and their actual intent. Learning directly from expert demonstrations could enable agents to better learn expert-curation strategies. This can be explored in two ways:

- Inverse Reinforcement Learning: One primary approach involves leveraging Inverse Reinforcement Learning (IRL), which addresses the reward design problem by inferring the reward function directly from observing expert demonstrations. IRL has been applied in previous works to learn sequential decisions [111]. In the context of playlist generation, the agent could learn directly from expert-curated sequences.
- Additional Prompting Techniques: A secondary approach involves exploring few-shot prompting techniques to improve the model's output quality and consistency, as demonstrated in the baseline Text2Reward paper [26]. Providing a few examples of expert-curated playlists alongside the expert interview transcripts could help the LLMs match expert descriptions with their existing work to learn expert strategies better.

8.1.4. Extensibility to Other Domains:

Wile this research focused exclusively on music, the core methodology has the potential for extensibility. Any field where success is defined by subjective and hard to codify expert knowledge could benefit from this approach of translating qualitative expertise into computable reward functions. In creative domains like narrative generation, the intuitive principles of storytelling could be elicited from expert authors to guide an RL agent in creating compelling plots. Similarly, in fields such as medical treatment planning, this framework could be leveraged to capture the decision-making processes of experienced clinicians, which are often difficult to define as simple rules.

References

[1] Markus Schedl et al. “Current challenges and visions in music recommender systems research”. In: *International Journal of Multimedia Information Retrieval* 7.2 (June 2018), pp. 95–116. ISSN: 2192-6611, 2192-662X. DOI: 10.1007/s13735-018-0154-2. URL: http://link.springer.com/10.1007/s13735-018-0154-2 (visited on 01/27/2025).

[2] *What Drives Demand for Playlists on Spotify?* DOI: 10.1287/mksc.2022.0273. URL: https://pubsonline.informs.org/doi/epdf/10.1287/mksc.2022.0273 (visited on 06/04/2025).

[3] Jack Webster. “The promise of personalisation: Exploring how music streaming platforms are shaping the performance of class identities and distinction”. In: *New Media & Society* 25.8 (2023), pp. 2140–2162.

[4] Brian J Hracs and Jack Webster. “From selling songs to engineering experiences: exploring the competitive strategies of music streaming platforms”. In: *Journal of Cultural Economy* 14.2 (2021), pp. 240–257.

[5] Fikra Ahnaf. “From listening to curating: Anthropological curatorship toward music playlist practices”. en. In: *ETNOSIA : Jurnal Etnografi Indonesia* 8.2 (Dec. 2023), pp. 271–289. ISSN: 2548-9747, 2527-9319. DOI: 10.31947/etnosia.v8i2.30162. URL: https://journal.unhas.ac.id/index.php/etnosia/article/view/30162 (visited on 06/02/2025).

[6] Ricardo Dias, Daniel Gonçalves, and Manuel J. Fonseca. “From manual to assisted playlist creation: a survey”. In: *Multimedia Tools and Applications* 76.12 (June 2017), pp. 14375–14403. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-016-3836-x. URL: http://link.springer.com/10.1007/s11042-016-3836-x (visited on 06/04/2025).

[7] Diana Lin and Sampath Jayarathna. “Automated Playlist Generation from Personal Music Libraries”. In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. July 2018, pp. 217–224. DOI: 10.1109/IRI.2018.00039. URL: https://ieeexplore.ieee.org/document/8424710/ (visited on 06/04/2025).

[8] Linnea Media. *AI Generated Playlists vs Curators: Who Wins?* 2024. URL: https://linnea.media/ai-generated-playlists-vs-curators/ (visited on 05/16/2024).

[9] Spotify Engineering. *Humans + Machines: A Look Behind the Playlists Powered by Spotify’s Algotorial Technology*. Spotify Engineering. Apr. 27, 2023. URL: https://engineering.atspotify.com/2023/04/humans-machines-a-look-behind-spotifys-algotorial-playlists/ (visited on 10/31/2024).

[10] Daily Playlists. *AI music curation versus Human music curation*. 2024. URL: https://dailyplaylists.com/en/blog/ai-music-curation-versus-human-music-curation (visited on 12/20/2024).

[11] Chaoguang Luo et al. *Against Filter Bubbles: Diversified Music Recommendation via Weighted Hypergraph Embedding Learning*. arXiv:2402.16299 [cs]. Feb. 2024. DOI: 10.48550/arXiv.2402.16299. URL: http://arxiv.org/abs/2402.16299 (visited on 06/04/2025).

[12] Qazi Mohammad Areeb et al. *Filter Bubbles in Recommender Systems: Fact or Fallacy – A Systematic Review*. arXiv:2307.01221 [cs]. July 2023. DOI: 10.48550/arXiv.2307.01221. URL: http://arxiv.org/abs/2307.01221 (visited on 06/04/2025).

[13] Natalie Clewley et al. “Eliciting Expert Knowledge to Inform Training Design”. In: *Proceedings of the 31st European Conference on Cognitive Ergonomics*. ECCE 2019: 31st European Conference on Cognitive Ergonomics. BELFAST United Kingdom: ACM, Sept. 10, 2019, pp. 138–143. ISBN: 978-1-4503-7166-7. DOI: 10.1145/3335082.3335091. URL: https://dl.acm.org/doi/10.1145/3335082.3335091 (visited on 11/19/2024).

[14] Wioleta Kucharska and G. Scott Erickson. “Tacit knowledge acquisition & sharing, and its influence on innovations: A Polish/US cross-country study”. en. In: *International Journal of Information Management* 71 (Aug. 2023), p. 102647. ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2023.102647. URL: https://linkinghub.elsevier.com/retrieve/pii/S0268401223000282 (visited on 06/09/2025).

[15] Ensiyeh Jamshidi et al. “How to utilize tacit knowledge in health organizations: An Iranian perspective”. In: *Medical Journal of the Islamic Republic of Iran* 32 (Nov. 22, 2018), p. 116. ISSN: 1016-1430. DOI: 10.14196/mjiri.32.116. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6387804/ (visited on 05/05/2025).

[16] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[17] Oleksandr D Rossiiev et al. “A comprehensive survey on reinforcement learning-based recommender systems: State-of-the-art, challenges, and future perspectives”. In: *CEUR Workshop Proceedings*. 2025, pp. 428–440.

[18] So Yeon Park and Blair Kaneshiro. “User perspectives on critical factors for collaborative playlists”. In: *PLOS ONE* 17.1 (Jan. 5, 2022). Ed. by Emőke-Ágnes Horvát, e0260750. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0260750. URL: https://dx.plos.org/10.1371/journal.pone.0260750 (visited on 11/14/2024).

[19] Y. V. Srinivasa Murthy and Shashidhar G. Koolagudi. “Content-Based Music Information Retrieval (CB-MIR) and Its Applications toward the Music Industry: A Review”. In: *ACM Computing Surveys* 51.3 (May 31, 2019), pp. 1–46. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3177849. URL: https://dl.acm.org/doi/10.1145/3177849 (visited on 03/05/2025).

[20] Xiangyu Zhao et al. *Deep Reinforcement Learning for List-wise Recommendations*. arXiv:1801.00209 [cs]. June 2019. DOI: 10.48550/arXiv.1801.00209. URL: http://arxiv.org/abs/1801.00209 (visited on 06/01/2025).

[21] Rishi Hazra et al. *REvolve: Reward Evolution with Large Language Models using Human Feedback*. Apr. 6, 2025. DOI: 10.48550/arXiv.2406.01309. arXiv: 2406.01309[cs]. URL: http://arxiv.org/abs/2406.01309 (visited on 05/05/2025).

[22] David Silver et al. “Reward is enough”. In: *Artificial Intelligence* 299 (Oct. 2021), p. 103535. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103535. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370221000862 (visited on 03/05/2025).

[23] Katharina Brennig. “Revealing the Unspoken: Using LLMs to Mobilize and Enrich Tacit Knowledge in Event Logs of Knowledge-Intensive Processes”. en. In: ().

[24] Raymond Li et al. *StarCoder: may the source be with you!* arXiv:2305.06161 [cs]. Dec. 2023. DOI: 10.48550/arXiv.2305.06161. URL: http://arxiv.org/abs/2305.06161 (visited on 06/09/2025).

[25] Hung Le et al. “CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning”. en. In: ().

[26] Tianbao Xie et al. “TE X T2RE W A R D : REWARD SHAPING WITH LANGUAGE MODELS FOR REINFORCEMENT LEARNING”. en. In: (2024).

[27] Minae Kwon et al. *Reward Design with Language Models*. Feb. 27, 2023. DOI: 10.48550/arXiv.2303.00001. arXiv: 2303.00001[cs]. URL: http://arxiv.org/abs/2303.00001 (visited on 12/16/2024).

[28] Geoffray Bonnin and Dietmar Jannach. “Automated Generation of Music Playlists: Survey and Experiments”. In: *ACM Computing Surveys* 47.2 (Jan. 8, 2015), pp. 1–35. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2652481. URL: https://dl.acm.org/doi/10.1145/2652481 (visited on 12/14/2024).

[29] Max J Pachali and Hannes Datta. “What drives demand for playlists on Spotify?” In: *Marketing Science* 44.1 (2025), pp. 54–64.

[30] Yuanguo Lin et al. “A Survey on Reinforcement Learning for Recommender Systems”. In: *IEEE Transactions on Neural Networks and Learning Systems* 35.10 (Oct. 2024), pp. 13164–13184. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2023.3280161. arXiv: 2109.10665[cs]. URL: http://arxiv.org/abs/2109.10665 (visited on 05/13/2025).

[31] Giovanni Gabbolini and Derek Bridge. “Surveying More Than Two Decades of Music Information Retrieval Research on Playlists”. In: *ACM Transactions on Intelligent Systems and Technology* (Aug. 12, 2024), p. 3688398. ISSN: 2157-6904, 2157-6912. DOI: 10.1145/3688398. URL: https://dl.acm.org/doi/10.1145/3688398 (visited on 11/04/2024).

[32] D. Gartner, F. Kraft, and T. Schaaf. “An Adaptive Distance Measure for Similarity Based Playlist Generation”. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07*. 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07. Vol. 1. ISSN: 2379-190X. Apr. 2007, pp. I–229–I–232. DOI: 10.1109/ICASSP.2007.366658. URL: https://ieeexplore.ieee.org/document/4217058/?arnumber=4217058 (visited on 03/05/2025).

[33] R. Ragno, C. J. C. Burges, and C. Herley. “Inferring similarity between music objects with application to playlist generation”. In: *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*. MM&Sec ’05: Multimedia and Security Workshop 2005. Hilton Singapore: ACM, Nov. 10, 2005, pp. 73–80. ISBN: 978-1-59593-244-0. DOI: 10.1145/1101826.1101840. URL: https://dl.acm.org/doi/10.1145/1101826.1101840 (visited on 03/05/2025).

[34] Francesco Ricci, Lior Rokach, and Bracha Shapira, eds. *Recommender Systems Handbook*. New York, NY: Springer US, 2022. ISBN: 978-1-07-162196-7 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4. URL: https://link.springer.com/10.1007/978-1-0716-2197-4 (visited on 01/27/2025).

[35] Elias Mann. “Context Aware Music Recommendation and Playlist Generation”. In: *SMU Journal of Undergraduate Research* 8.2 (May 2024). DOI: 10.25172/jour.8.2.1. URL: https://scholar.smu.edu/jour/vol8/iss2/2/ (visited on 01/27/2025).

[36] Shuiguang Deng et al. “Exploring user emotion in microblogs for music recommendation”. In: *Expert Systems with Applications* 42.23 (Dec. 2015), pp. 9284–9293. ISSN: 09574174. DOI: 10.1016/j.eswa.2015.08.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417415005746 (visited on 01/27/2025).

[37] Maake Benard Magara et al. “MPlist: Context aware music playlist”. In: *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*. 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech). Aug. 2016, pp. 309–316. DOI: 10.1109/EmergiTech.2016.7737358. URL: https://ieeexplore.ieee.org/document/7737358/?arnumber=7737358 (visited on 01/27/2025).

[38] Perfecto Herrera, Zuriñe Resa, and Mohamed Sordo. “Rocking around the clock eight days a week: an exploration of temporal patterns of music listening”. In: ().

[39] Shun-Yao Shih and Heng-Yu Chi. *Automatic, Personalized, and Flexible Playlist Generation using Reinforcement Learning*. arXiv:1809.04214 [cs]. Sept. 2018. DOI: 10.48550/arXiv.1809.04214. URL: http://arxiv.org/abs/1809.04214 (visited on 12/10/2024).

[40] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. “Deep content-based music recommendation”. In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013. URL: https://papers.nips.cc/paper_files/paper/2013/hash/b3ba8f1bee1238a2f37603d90b58898d-Abstract.html (visited on 01/21/2025).

[41] Matej Bevec, Marko Tkalčič, and Matevž Pesek. “Hybrid music recommendation with graph neural networks”. en. In: *User Modeling and User-Adapted Interaction* 34.5 (Nov. 2024), pp. 1891–1928. ISSN: 0924-1868, 1573-1391. DOI: 10.1007/s11257-024-09410-4. URL: https://link.springer.com/10.1007/s11257-024-09410-4 (visited on 05/28/2025).

[42] Enrico Palumbo et al. “Knowledge Graph Embeddings with node2vec for Item Recommendation”. In: *The Semantic Web: ESWC 2018 Satellite Events*. European Semantic Web Conference. ISSN: 1611-3349. Springer, Cham, 2018, pp. 117–120. ISBN: 978-3-319-98192-5. DOI: 10.1007/978-3-319-98192-5_22. URL: https://link.springer.com/chapter/10.1007/978-3-319-98192-5_22 (visited on 05/19/2025).

[43] Aditya Grover and Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. arXiv:1607.00653 [cs] version: 1. July 2016. DOI: 10.48550/arXiv.1607.00653. URL: http://arxiv.org/abs/1607.00653 (visited on 05/19/2025).

[44] Balázs Hidasi et al. *Session-based Recommendations with Recurrent Neural Networks*. Mar. 29, 2016. DOI: 10.48550/arXiv.1511.06939. arXiv: 1511.06939[cs]. URL: http://arxiv.org/abs/1511.06939 (visited on 04/10/2025).

[45] *“First Week Is Editorial, Second Week Is Algorithmic”: Platform Gatekeepers and the Platformization of Music Curation*. en. DOI: 10.1177/2056305119880006. URL: https://journals.sagepub.com/doi/epub/10.1177/2056305119880006 (visited on 04/21/2025).

[46] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. “‘More of an Art than a Science’: Supporting the Creation of Playlists and Mixes”. en. In: ().

[47] Maria Eriksson. “The editorial playlist as container technology: on Spotify and the logistical role of digital music packages”. In: *Journal of Cultural Economy* 13.4 (July 2020). Publisher: Routledge _eprint: https://doi.org/10.1080/17530350.2019.1708780, pp. 415–427. ISSN: 1753-0350. DOI: 10.1080/17530350.2019.1708780. URL: https://doi.org/10.1080/17530350.2019.1708780 (visited on 01/21/2025).

[48] Shobu Ikeda, Kenta Oku, and Kyoji Kawagoe. “Music Playlist Recommendation Using Acoustic-Feature Transition Inside the Songs”. In: *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia - MoMM2017*. the 15th International Conference. Salzburg, Austria: ACM Press, 2017, pp. 216–219. ISBN: 978-1-4503-5300-7. DOI: 10.1145/3151848.3151880. URL: http://dl.acm.org/citation.cfm?doid=3151848.3151880 (visited on 04/21/2025).

[49] Pedro A. S. O. Neto et al. “The algorithmic nature of song-sequencing: statistical regularities in music albums”. In: *Journal of New Music Research* 0.0 (Nov. 2024). Publisher: Routledge _eprint: https://doi.org/10.1080/09298215.2024.2423610, pp. 1–15. ISSN: 0929-8215. DOI: 10.1080/09298215.2024.2423610. URL: https://doi.org/10.1080/09298215.2024.2423610 (visited on 02/26/2025).

[50] Andreu Vall et al. *The Importance of Song Context and Song Order in Automated Music Playlist Generation*. arXiv:1807.04690 [cs]. July 2018. DOI: 10.48550/arXiv.1807.04690. URL: http://arxiv.org/abs/1807.04690 (visited on 04/10/2025).

[51] Harald Schweiger, Emilia Parada-Cabaleiro, and Markus Schedl. “DOES TRACK SEQUENCE IN USER-GENERATED PLAYLISTS MATTER?” In: (2021).

[52] Ching-Wei Chen et al. “Recsys challenge 2018: automatic music playlist continuation”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys ’18: Twelfth ACM Conference on Recommender Systems. Vancouver British Columbia Canada: ACM, Sept. 27, 2018, pp. 527–528. ISBN: 978-1-4503-5901-6. DOI: 10.1145/3240323.3240342. URL: https://dl.acm.org/doi/10.1145/3240323.3240342 (visited on 01/13/2025).

[53] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. “Facing the cold start problem in recommender systems”. en. In: *Expert Systems with Applications* 41.4 (Mar. 2014), pp. 2065–2073. ISSN: 09574174. DOI: 10.1016/j.eswa.2013.09.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417413007240 (visited on 06/10/2025).

[54] Ali Yürekli, Cihan Kaleli, and Alper Bilge. “Alleviating the cold-start playlist continuation in music recommendation using latent semantic indexing”. en. In: *International Journal of Multimedia Information Retrieval* 10.3 (Sept. 2021), pp. 185–198. ISSN: 2192-6611, 2192-662X. DOI: 10.1007/s13735-021-00214-5. URL: https://link.springer.com/10.1007/s13735-021-00214-5 (visited on 02/04/2025).

[55] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. Dec. 5, 2017. DOI: 10.48550/arXiv.1712.01815. arXiv: 1712.01815[cs]. URL: http://arxiv.org/abs/1712.01815 (visited on 03/05/2025).

[56] Florian Fuchs et al. "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 6.3 (July 2021). Conference Name: IEEE Robotics and Automation Letters, pp. 4257–4264. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3064284. URL: https://ieeexplore.ieee.org/document/9372847/?arnumber=9372847 (visited on 03/05/2025).

[57] Kenneth Schröder, Alexander Kastius, and Rainer Schlosser. "Reinforcement Learning Algorithms: Categorization and Structural Properties". In: *International Conference on Operations Research and Enterprise Systems*. Springer. 2022, pp. 96–120.

[58] OpenAI. *Kinds of RL Algorithms*. 2018. URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.

[59] Gabriel Dulac-Arnold et al. *Deep Reinforcement Learning in Large Discrete Action Spaces*. arXiv:1512.07679. Apr. 2016. URL: http://arxiv.org/abs/1512.07679 (visited on 11/21/2024).

[60] Yujing Hu et al. *Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application*. arXiv:1803.00710 [cs]. May 2018. DOI: 10.48550/arXiv.1803.00710. URL: http://arxiv.org/abs/1803.00710 (visited on 06/09/2025).

[61] John Schulman et al. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347 [cs]. Aug. 2017. DOI: 10.48550/arXiv.1707.06347. URL: http://arxiv.org/abs/1707.06347 (visited on 06/09/2025).

[62] Vaibhav Padhye, Kailasam Lakshmanan, and Amrita Chaturvedi. "Proximal policy optimization based hybrid recommender systems for large scale recommendations". In: *Multimedia Tools and Applications* 82.13 (May 2023), pp. 20079–20100. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-022-14231-x. URL: https://link.springer.com/10.1007/s11042-022-14231-x (visited on 06/09/2025).

[63] Gabriel Kalweit et al. "Q-learning with Long-term Action-space Shaping to Model Complex Behavior for Autonomous Lane Changes". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866. Sept. 2021, pp. 5641–5648. DOI: 10.1109/IROS51168.2021.9636668. URL: https://ieeexplore.ieee.org/document/9636668/?arnumber=9636668 (visited on 03/04/2025).

[64] Cheng-Yen Tang et al. "Implementing action mask in proximal policy optimization (PPO) algorithm". In: *ICT Express* 6.3 (Sept. 2020), pp. 200–203. ISSN: 24059595. DOI: 10.1016/j.icte.2020.05.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405959520300746 (visited on 03/04/2025).

[65] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. *Action Space Shaping in Deep Reinforcement Learning*. May 26, 2020. DOI: 10.48550/arXiv.2004.00980. arXiv: 2004.00980[cs]. URL: http://arxiv.org/abs/2004.00980 (visited on 03/04/2025).

[66] Shengyi Huang and Santiago Ontañón. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms". In: *The International FLAIRS Conference Proceedings* 35 (May 4, 2022). ISSN: 2334-0762. DOI: 10.32473/flairs.v35i.130584. arXiv: 2006.14171[cs]. URL: http://arxiv.org/abs/2006.14171 (visited on 05/12/2025).

[67] Deheng Ye et al. *Mastering Complex Control in MOBA Games with Deep Reinforcement Learning*. Dec. 15, 2020. DOI: 10.48550/arXiv.1912.09729. arXiv: 1912.09729[cs]. URL: http://arxiv.org/abs/1912.09729 (visited on 03/04/2025).

[68] Oriol Vinyals et al. *StarCraft II: A New Challenge for Reinforcement Learning*. Aug. 16, 2017. DOI: 10.48550/arXiv.1708.04782. arXiv: 1708.04782[cs]. URL: http://arxiv.org/abs/1708.04782 (visited on 03/04/2025).

[69] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: ().

[70] Marcin Andrychowicz et al. “Hindsight Experience Replay”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html (visited on 04/10/2025).

[71] Sven Koenig and Reid G Simmons. “The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms”. In: *Machine Learning* 22.1 (1996), pp. 227–250.

[72] Michael Polanyi. “The Logic of Tacit Inference”. en. In: *Philosophy* 41.155 (Jan. 1966), pp. 1–18. ISSN: 0031-8191, 1469-817X. DOI: 10.1017/S0031819100066110. URL: https://www.cambridge.org/core/product/identifier/S0031819100066110/type/journal_article (visited on 06/09/2025).

[73] Mohammad Nazim and Bhaskar Mukherjee. “Knowledge Management Strategy”. en. In: *Knowledge Management in Libraries*. Elsevier, 2016, pp. 89–113. ISBN: 978-0-08-100564-4. DOI: 10.1016/B978-0-08-100564-4.00005-3. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780081005644000053 (visited on 06/09/2025).

[74] Omolola A. Adeoye-Olatunde and Nicole L. Olenik. “Research and scholarly methods: Semi-structured interviews”. en. In: *JACCP: JOURNAL OF THE AMERICAN COLLEGE OF CLINICAL PHARMACY* 4.10 (2021). _eprint: https://accpjournals.onlinelibrary.wiley.com/doi/pdf/10.1002/jac5.1441, pp. 1358–1367. ISSN: 2574-9870. DOI: 10.1002/jac5.1441. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jac5.1441 (visited on 05/05/2025).

[75] Shan Qiao et al. “Generative AI for thematic analysis in a maternal health study: coding semistructured interviews using large language models”. In: *Applied Psychology: Health and Well-Being* 17.3 (June 1, 2025). Publisher: John Wiley & Sons, Ltd, e70038. ISSN: 1758-0854. DOI: 10.1111/aphw.70038. URL: https://iaap.journals.onlinelibrary.wiley.com/doi/10.1111/aphw.70038 (visited on 06/05/2025).

[76] Samuel Kernan Freire et al. “Tacit Knowledge Elicitation for Shop-floor Workers with an Intelligent Assistant”. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI ’23: CHI Conference on Human Factors in Computing Systems. Hamburg Germany: ACM, Apr. 19, 2023, pp. 1–7. ISBN: 978-1-4503-9422-2. DOI: 10.1145/3544549.3585755. URL: https://dl.acm.org/doi/10.1145/3544549.3585755 (visited on 05/16/2025).

[77] Emilia Rosselli Del Turco and Peter Dalsgaard. ““My ideas come little by little”: how graphic professionals manage ideas”. en. In: *Creativity and Cognition*. Chicago IL USA: ACM, June 2024, pp. 452–463. ISBN: 979-8-4007-0485-7. DOI: 10.1145/3635636.3656203. URL: https://dl.acm.org/doi/10.1145/3635636.3656203 (visited on 06/04/2025).

[78] Kihoon Son et al. *Demystifying Tacit Knowledge in Graphic Design: Characteristics, Instances, Approaches, and Guidelines*. Mar. 10, 2024. DOI: 10.48550/arXiv.2403.06252. arXiv: 2403.06252[cs]. URL: http://arxiv.org/abs/2403.06252 (visited on 05/05/2025).

[79] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 06/09/2025).

[80] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv:1910.10683 [cs]. Sept. 2023. DOI: 10.48550/arXiv.1910.10683. URL: http://arxiv.org/abs/1910.10683 (visited on 06/09/2025).

[81] Humza Naveed et al. *A Comprehensive Overview of Large Language Models*. arXiv:2307.06435 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2307.06435. URL: http://arxiv.org/abs/2307.06435 (visited on 06/09/2025).

[82] OpenAI et al. *GPT-4 Technical Report*. arXiv:2303.08774 [cs]. Mar. 2024. DOI: 10.48550/arXiv.2303.08774. URL: http://arxiv.org/abs/2303.08774 (visited on 06/09/2025).

[83] Gemini Team et al. *Gemini: A Family of Highly Capable Multimodal Models*. May 9, 2025. DOI: 10.48550/arXiv.2312.11805. arXiv: 2312.11805[cs]. URL: http://arxiv.org/abs/2312.11805 (visited on 06/09/2025).

[84] Anthropic Claude. *Title of the Website*. 2024. URL: https://www.anthropic.com/claude.

[85] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. arXiv:2107.03374 [cs]. July 2021. DOI: 10.48550/arXiv.2107.03374. URL: http://arxiv.org/abs/2107.03374 (visited on 06/09/2025).

[86] Junkai Chen et al. *NLPerturbator: Studying the Robustness of Code LLMs to Natural Language Variations*. arXiv:2406.19783 [cs]. June 2024. DOI: 10.48550/arXiv.2406.19783. URL: http://arxiv.org/abs/2406.19783 (visited on 06/09/2025).

[87] Tomislav Duricic et al. "Empirical Comparison of Graph Embeddings for Trust-Based Collaborative Filtering". In: *Foundations of Intelligent Systems*. International Symposium on Methodologies for Intelligent Systems. ISSN: 1611-3349. Springer, Cham, 2020, pp. 181–191. ISBN: 978-3-030-59491-6. DOI: 10.1007/978-3-030-59491-6_17. URL: https://link.springer.com/chapter/10.1007/978-3-030-59491-6_17 (visited on 05/19/2025).

[88] Janez Demsˇar and Janez Demsar. "Statistical Comparisons of Classifiers over Multiple Data Sets". en. In: ().

[89] *Quick and Easy Implementation of the Benjamini-Hochberg Procedure for Controlling the False Positive Rate in Multiple Comparisons - David Thissen, Lynne Steinberg, Daniel Kuang, 2002*. URL: https://journals-sagepub-com.tudelft.idm.oclc.org/doi/abs/10.3102/10769986027001077?casa_token=iXZlZw5uF_gAAAAA:K5PiMxIskYFwz_NWqhpBKOFFf9ne3Uc383WWpkrPiq9emBhrbetwxySHcDXeZ3nBxFqhOn25iBr-11k (visited on 07/23/2025).

[90] Federico Tomasi et al. "Automatic Music Playlist Generation via Simulation-based Reinforcement Learning". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '23: The 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Long Beach CA USA: ACM, Aug. 6, 2023, pp. 4948–4957. ISBN: 9798400701030. DOI: 10.1145/3580305.3599777. URL: https://dl.acm.org/doi/10.1145/3580305.3599777 (visited on 11/21/2024).

[91] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. *DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation*. arXiv:1401.1880 [cs]. Mar. 2015. DOI: 10.48550/arXiv.1401.1880. URL: http://arxiv.org/abs/1401.1880 (visited on 04/21/2025).

[92] Keigo Sakurai et al. "Controllable Music Playlist Generation Based on Knowledge Graph and Reinforcement Learning". In: *Sensors* 22.10 (May 13, 2022), p. 3722. ISSN: 1424-8220. DOI: 10.3390/s22103722. URL: https://www.mdpi.com/1424-8220/22/10/3722 (visited on 12/16/2024).

[93] James King and Vaiva Imbrasaitė. "Generating Music Playlists with Hierarchical Clustering and Q-Learning". In: *Advances in Information Retrieval*. Ed. by Allan Hanbury et al. Vol. 9022. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 315–326. ISBN: 978-3-319-16353-6 978-3-319-16354-3. DOI: 10.1007/978-3-319-16354-3_34. URL: http://link.springer.com/10.1007/978-3-319-16354-3_34 (visited on 11/26/2024).

[94] Keigo Sakurai et al. "Music Playlist Generation Based on Graph Exploration Using Reinforcement Learning". In: *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*. Mar. 2021, pp. 53–54. DOI: 10.1109/LifeTech52111.2021.9391870. URL: https://ieeexplore.ieee.org/document/9391870/ (visited on 06/06/2025).

[95] Jaehun Kim et al. "Towards Seed-Free Music Playlist Generation: Enhancing Collaborative Filtering with Playlist Title Information". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge '18: Proceedings of the ACM Recommender Systems Challenge 2018. Vancouver BC Canada: ACM, Oct. 2, 2018, pp. 1–6. ISBN: 978-1-4503-6586-4. DOI: 10.1145/3267471.3267485. URL: https://dl.acm.org/doi/10.1145/3267471.3267485 (visited on 11/08/2024).

[96] Alec Nonnemaker. "Enabling Targeted Music Exploration with Interactive Recommendations". In: ().

[97] Matthew C. McCallum et al. *Supervised and Unsupervised Learning of Audio Representations for Music Understanding*. arXiv:2210.03799 [cs]. Oct. 2022. DOI: 10.48550/arXiv.2210.03799. URL: http://arxiv.org/abs/2210.03799 (visited on 01/13/2025).

[98] Binbin Hu, Chuan Shi, and Jian Liu. “Playlist Recommendation Based on Reinforcement Learning”. en. In: *Intelligence Science I*. Ed. by Zhongzhi Shi, Ben Goertzel, and Jiali Feng. Cham: Springer International Publishing, 2017, pp. 172–182. ISBN: 978-3-319-68121-4. DOI: 10.1007/978-3-319-68121-4_18.

[99] Keigo Sakurai et al. “Music Playlist Generation Based on Reinforcement Learning Using Acoustic Feature Map”. In: *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*. ISSN: 2378-8143. Oct. 2020, pp. 942–943. DOI: 10.1109/GCCE50665.2020.9291748. URL: https://ieeexplore.ieee.org/document/9291748/ (visited on 04/21/2025).

[100] Adam S. Hayes. ““Conversing” With Qualitative Data: Enhancing Qualitative Research Through Large Language Models (LLMs)”. In: *International Journal of Qualitative Methods* 24 (Dec. 2025), p. 16094069251322346. ISSN: 1609-4069, 1609-4069. DOI: 10.1177/16094069251322346. URL: https://journals.sagepub.com/doi/10.1177/16094069251322346 (visited on 03/05/2025).

[101] Manya Wadhwa et al. *Using Natural Language Explanations to Rescale Human Judgments*. Sept. 9, 2024. DOI: 10.48550/arXiv.2305.14770. arXiv: 2305.14770[cs]. URL: http://arxiv.org/abs/2305.14770 (visited on 03/05/2025).

[102] Valentin Ritschl, Lars Grespan, and Datum Unterschrift Ort. “Assessing the Effectiveness of Large Language Models in Qualitative Content Analysis of Interview Data”. In: ().

[103] Jiayang Li and Jiale Li. *A Map of Exploring Human Interaction patterns with LLM: Insights into Collaboration and Creativity*. Apr. 6, 2024. DOI: 10.48550/arXiv.2404.04570. arXiv: 2404.04570[cs]. URL: http://arxiv.org/abs/2404.04570 (visited on 03/06/2025).

[104] Kien Nguyen-Trung. “ChatGPT in thematic analysis: Can AI become a research assistant in qualitative research?” en. In: *Quality & Quantity* (June 2025). ISSN: 0033-5177, 1573-7845. DOI: 10.1007/s11135-025-02165-z. URL: https://link.springer.com/10.1007/s11135-025-02165-z (visited on 06/05/2025).

[105] Theo X. Olausson et al. *Is Self-Repair a Silver Bullet for Code Generation?* arXiv:2306.09896 [cs] version: 5. Feb. 2024. DOI: 10.48550/arXiv.2306.09896. URL: http://arxiv.org/abs/2306.09896 (visited on 06/05/2025).

[106] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Anchorage AK USA: ACM, July 25, 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701. URL: https://dl.acm.org/doi/10.1145/3292500.3330701 (visited on 07/14/2025).

[107] Lucien Heitz et al. “Deliberative Diversity for News Recommendations: Operationalization and Experimental User Study”. In: *Proceedings of the 17th ACM Conference on Recommender Systems*. Singapore Singapore: ACM, Sept. 2023, pp. 813–819. DOI: 10.1145/3604915.3608834. URL: https://dl.acm.org/doi/10.1145/3604915.3608834 (visited on 07/20/2025).

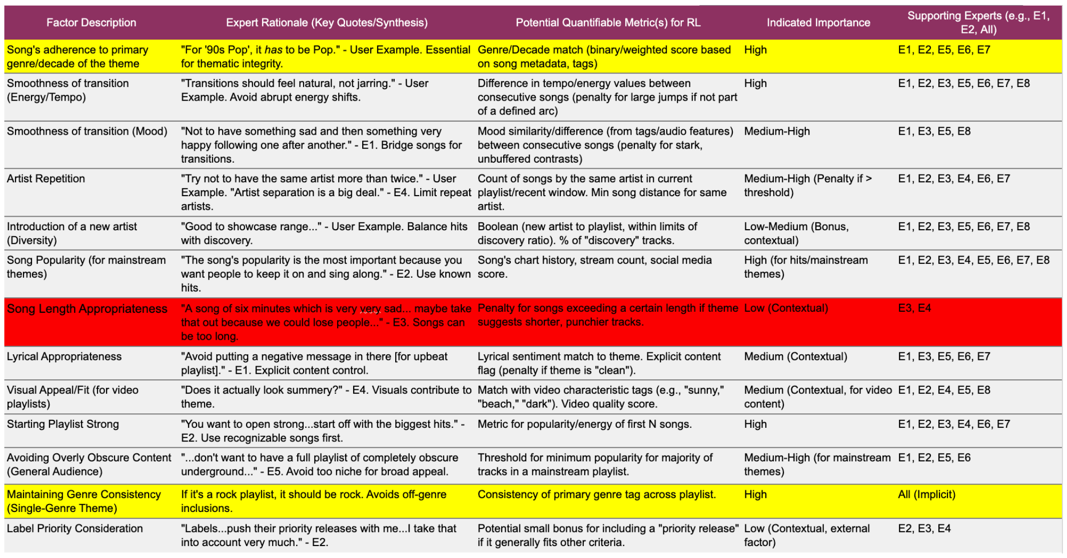
[108] Brett Binst. “How to Evaluate Serendipity in Recommender Systems: the Need for a Serendiptionnaire”. In: *18th ACM Conference on Recommender Systems*. Bari Italy: ACM, Oct. 2024, pp. 1335–1341. DOI: 10.1145/3640457.3688017. URL: https://dl.acm.org/doi/10.1145/3640457.3688017 (visited on 07/20/2025).

[109] Walid Bendada et al. *A Scalable Framework for Automatic Playlist Continuation on Music Streaming Services*. Apr. 12, 2023. arXiv: 2304.09061. URL: http://arxiv.org/abs/2304.09061 (visited on 10/31/2024).

[110] Hei-Chia Wang, Sheng-Wei Syu, and Papis Wongchaisuwat. “A method of music autotagging based on audio and lyrics”. In: *Multimedia Tools and Applications* 80.10 (Apr. 2021), pp. 15511–15539. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-020-10381-y. URL: https://link.springer.com/10.1007/s11042-020-10381-y (visited on 01/21/2025).

[111] Siyuan Liu et al. “Understanding Sequential Decisions via Inverse Reinforcement Learning”. In: *2013 IEEE 14th International Conference on Mobile Data Management*. 2013 IEEE 14th International Conference on Mobile Data Management. Vol. 1. ISSN: 2375-0324. June 2013, pp. 177–186. DOI: 10.1109/MDM.2013.28. URL: https://ieeexplore.ieee.org/abstract/document/6569134 (visited on 10/29/2024).

A
Qualitative Analysis of Interview Transcripts



Factor Description	Expert Rationale (Key Quotes/Synthesis)	Potential Quantifiable Metric(s) for RL	Indicated Importance	Supporting Experts (e.g., E1, E2, All)
Song's adherence to primary genre/decade of the theme	"For '90s Pop', it *has* to be Pop." - User Example. Essential for thematic integrity.	Genre/Decade match (binary/weighted score based on song metadata, tags)	High	E1, E2, E5, E6, E7
Smoothness of transition (Energy/Tempo)	"Transitions should feel natural, not jarring." - User Example. Avoid abrupt energy shifts.	Difference in tempo/energy values between consecutive songs (penalty for large jumps if not part of a defined arc)	High	E1, E2, E3, E5, E6, E7, E8
Smoothness of transition (Mood)	"Not to have something sad and then something very happy following one after another." - E1. Bridge songs for transitions.	Mood similarity/difference (from tags/audio features) between consecutive songs (penalty for stark, unbuffered contrasts)	Medium-High	E1, E3, E5, E8
Artist Repetition	"Try not to have the same artist more than twice." - User Example. "Artist separation is a big deal." - E4. Limit repeat artists.	Count of songs by the same artist in current playlist/recent window. Min song distance for same artist.	Medium-High (Penalty if > threshold)	E1, E2, E3, E4, E6, E7
Introduction of a new artist (Diversity)	"Good to showcase range..." - User Example. Balance hits with discovery.	Boolean (new artist to playlist, within limits of discovery ratio). % of "discovery" tracks.	Low-Medium (Bonus, contextual)	E1, E2, E3, E5, E6, E7, E8
Song Popularity (for mainstream themes)	"The song's popularity is the most important because you want people to keep it on and sing along." - E2. Use known hits.	Song's chart history, stream count, social media score.	High (for hits/mainstream themes)	E1, E2, E3, E4, E5, E6, E7, E8
Song Length Appropriateness	"A song of six minutes which is very very sad... maybe take that out because we could lose people..." - E3. Songs can be too long.	Penalty for songs exceeding a certain length if theme suggests shorter, punchier tracks.	Low (Contextual)	E3, E4
Lyrical Appropriateness	"Avoid putting a negative message in there [for upbeat playlist]." - E1. Explicit content control.	Lyrical sentiment match to theme. Explicit content flag (penalty if theme is "clean").	Medium (Contextual)	E1, E3, E5, E6, E7
Visual Appeal/Fit (for video playlists)	"Does it actually look summery?" - E4. Visuals contribute to theme.	Match with video characteristic tags (e.g., "sunny," "beach," "dark"). Video quality score.	Medium (Contextual, for video content)	E1, E2, E4, E5, E8
Starting Playlist Strong	"You want to open strong...start off with the biggest hits." - E2. Use recognizable songs first.	Metric for popularity/energy of first N songs.	High	E1, E2, E3, E4, E6, E7
Avoiding Overly Obscure Content (General Audience)	"...don't want to have a full playlist of completely obscure underground..." - E5. Avoid too niche for broad appeal.	Threshold for minimum popularity for majority of tracks in a mainstream playlist.	Medium-High (for mainstream themes)	E1, E2, E5, E6
Maintaining Genre Consistency (Single-Genre Theme)	If it's a rock playlist, it should be rock. Avoids off-genre inclusions.	Consistency of primary genre tag across playlist.	High	All (Implicit)
Label Priority Consideration	"Labels...push their priority releases with me...I take that into account very much." - E2.	Potential small bonus for including a "priority release" if it generally fits other criteria.	Low (Contextual, external factor)	E2, E3, E4

Figure A.1: Summarization of the Interview Transcripts by LLM (Gemini)

B
Prompt

The following prompt was provided to the language model to generate the reward function, its components, weights, and Python implementation.

You are an expert in Reinforcement Learning. We are developing a Reinforcement Learning (RL) agent to generate theme-based music playlists. The goal is for the RL agent (using Proximal Policy Optimization - PPO) to generate playlists that align with the creative decision-making processes and preferences of expert music curators at the company.

Available Data & Environment Setup: *environment details*

Formulate a Dense Reward Function Based on the provided expert interview transcripts,(8 experts in playlist curation were interviewed) your task is to:

1. Define a Dense Reward Function Formula: The reward function should be designed based on the inputs from the experts. This formula should assign a numerical reward at each step (i.e., for each song added to the playlist).
2. Assign Weights to Factors: Propose a set of weights for each component in your reward function. Explain the rationale behind the components and weights, considering the relative importance of different factors as implied by the expert insights.
3. Provide Python Code: Implement the proposed reward function in Python. The function should take the current state (e.g., current playlist, playlist title, candidate songs) and the chosen action (song added) as input. The code should be in a rewarder class format, similar to shown below: *Python code*

Key Considerations for the Reward Function:

- Density: The reward should be given at each step (each song added).
- Use of Embeddings: Leverage the Node2Vec embeddings.

Output Format:

1. Reward Function Formula:
2. Explanation of Components and Weights:
3. Python Code Implementation:

C Hyperparameter Details

In this section, we provide the details of the hyperparameter configuration used for our Reinforcement Learning (RL) agent.

For the RL training, we used an open-source implementation of the Proximal Policy Optimization (PPO) algorithm. To ensure a fair and controlled comparison between the different reward functions, we first performed a systematic hyperparameter search using the Optuna framework on our baseline manual reward model (the agent trained with the manually defined reward function). The search was conducted over 30 trials with the objective of maximizing the mean reward. The results of the top 10 performing trials are detailed in Table C.1.

Reward Score	Batch Size	Clip Range	Gamma	Learning Rate	Epochs	Steps
94.84546	128	0.25358	0.99447	4e-5	8	512
93.23071	128	0.13383	0.98920	4.8e-4	8	1024
92.37172	128	0.10260	0.98433	9e-5	11	2048
91.66009	128	0.16469	0.96468	1.1e-4	8	1024
88.85317	128	0.21303	0.97754	8e-5	7	2048
88.35233	128	0.25063	0.93344	2.8e-4	8	512
87.30213	128	0.12839	0.97671	1.1e-4	12	1024
87.25313	32	0.26090	0.98337	2e-5	16	2048
87.03858	128	0.11859	0.96334	3e-5	10	512
85.73132	128	0.17531	0.98871	1.6e-4	9	512

Table C.1: Top 10 Hyperparameter Configurations from Optuna Study

The single best-performing configuration from this search was then selected and used consistently for training all models throughout this study. The final, fixed hyperparameter values used for all experiments are listed in Table C.2.

Hyperparameter	Final Value
Discount Factor (γ)	0.99447
PPO Clip Range	0.25358
Learning Rate	4e-5
Number of Epochs	8
Number of Steps per Update	512
Batch Size	128

Table C.2: Final Hyperparameter Configuration used for training the RL agents

D Training and Performance Stability of RL Agents

This appendix provides detailed evidence of the training process for the RL agents. All experiments were conducted in the playlist generation environment as described in the main report.

Following the protocol outlined in Section 4.2.2, we trained each RL agent five times with different random seeds to ensure our results are reliable and not due to stochastic chance. We show this through the learning curve plots of the RL agents during training. Furthermore, we evaluate and compare the scores of each instance of the RL agent to ensure the results obtained are consistent enough to be combined for a given agent.

D.1. Learning Curves of RL-M

This section shows the learning curves for the agent trained on the manually-defined reward function (RL-M). The plots display the mean reward component scores and standard deviation across five runs over 5 million training steps. The learning curves for each agent, demonstrate consistent convergence with low variance across the independent runs for the key reward components of that agent. The narrow standard deviation bands in the performance plots (Figures D.1 - D.7) confirm that the training process was stable, especially in the convergence of the overall reward.

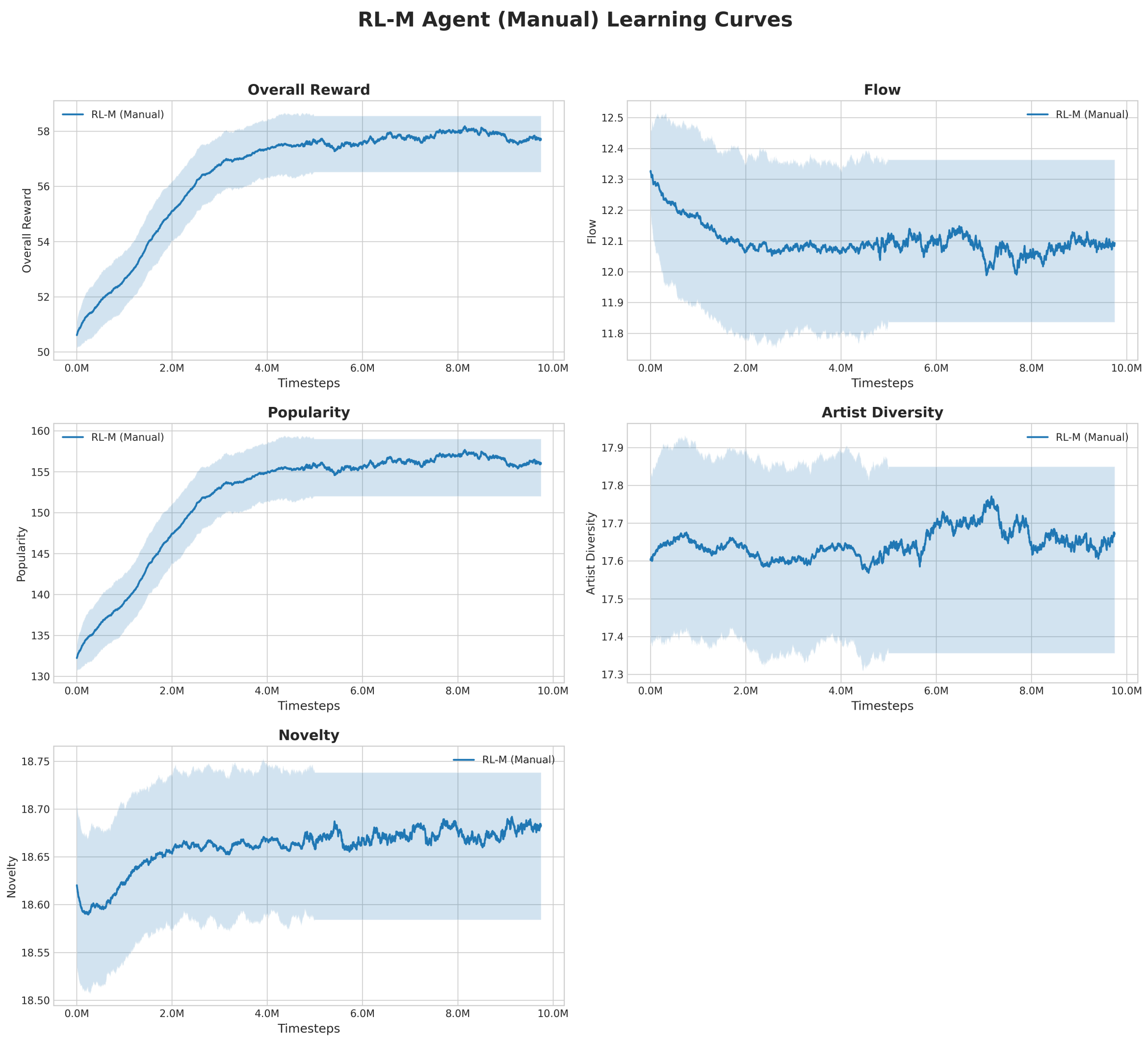


Figure D.1: Learning curves for the agent trained with the manually-defined reward function(RL-M). The agent shows consistent learning across all reward components, establishing a strong baseline for comparison. The low variance indicates a highly stable and reproducible training process.

D.2. Pipeline A: RL-LLM-S

The figures in this section show the learning curves for agents trained on reward functions generated from summarized interview transcripts. Each figure corresponds to a policy trained using a different LLM.

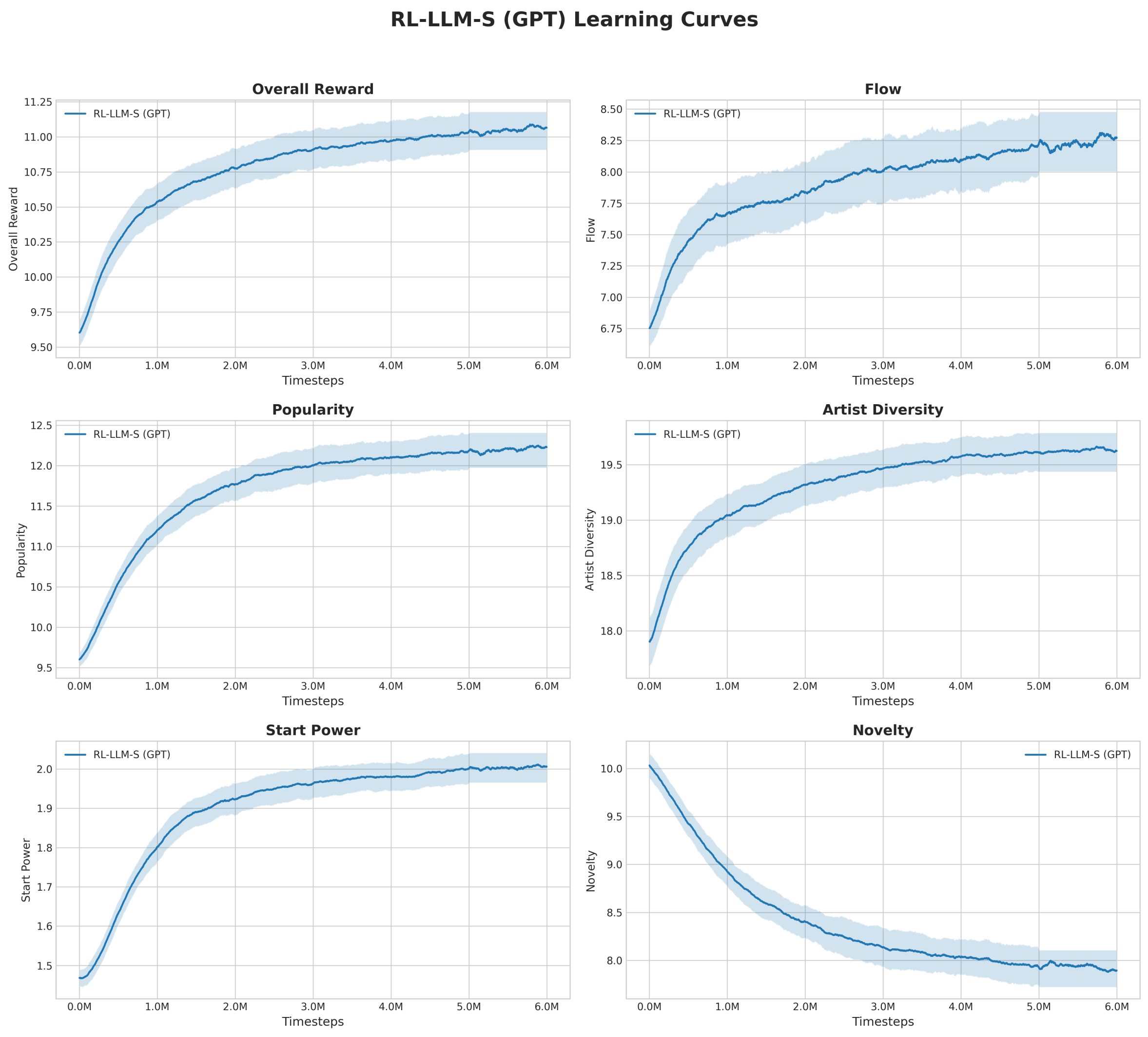


Figure D.2: Learning curves for the policy guided by rewards from **GPT-4o (Summarized)**. The agent demonstrates stable convergence, particularly in Overall Reward and Flow, indicating that summarized reviews provide a reliable signal for policy optimization.

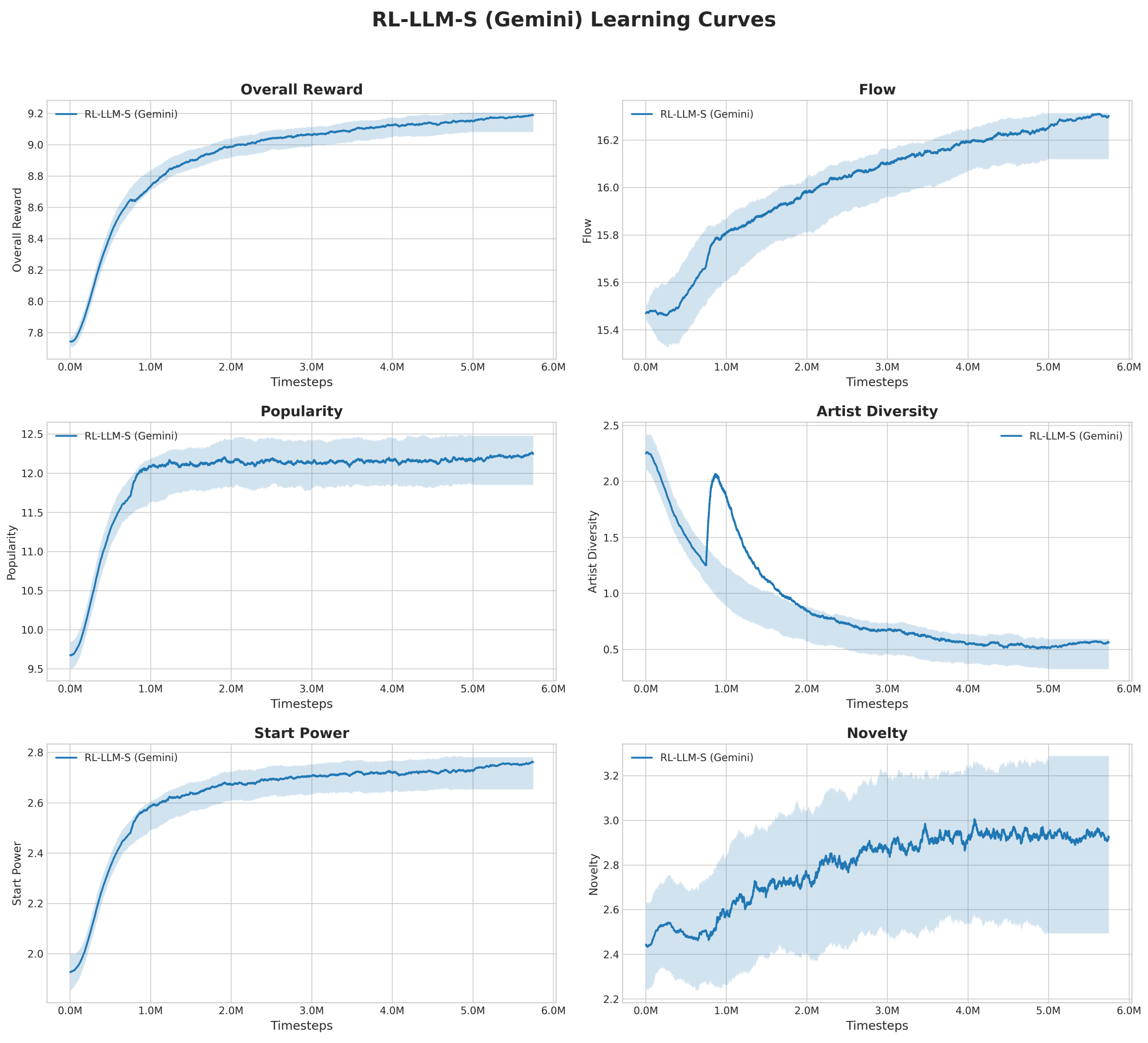


Figure D.3: Learning curves for the policy guided by rewards from **Gemini 2.5 Pro (Summarized)**. The training process is stable, with tight standard deviation bands across most metrics, confirming the reliability of the agent.

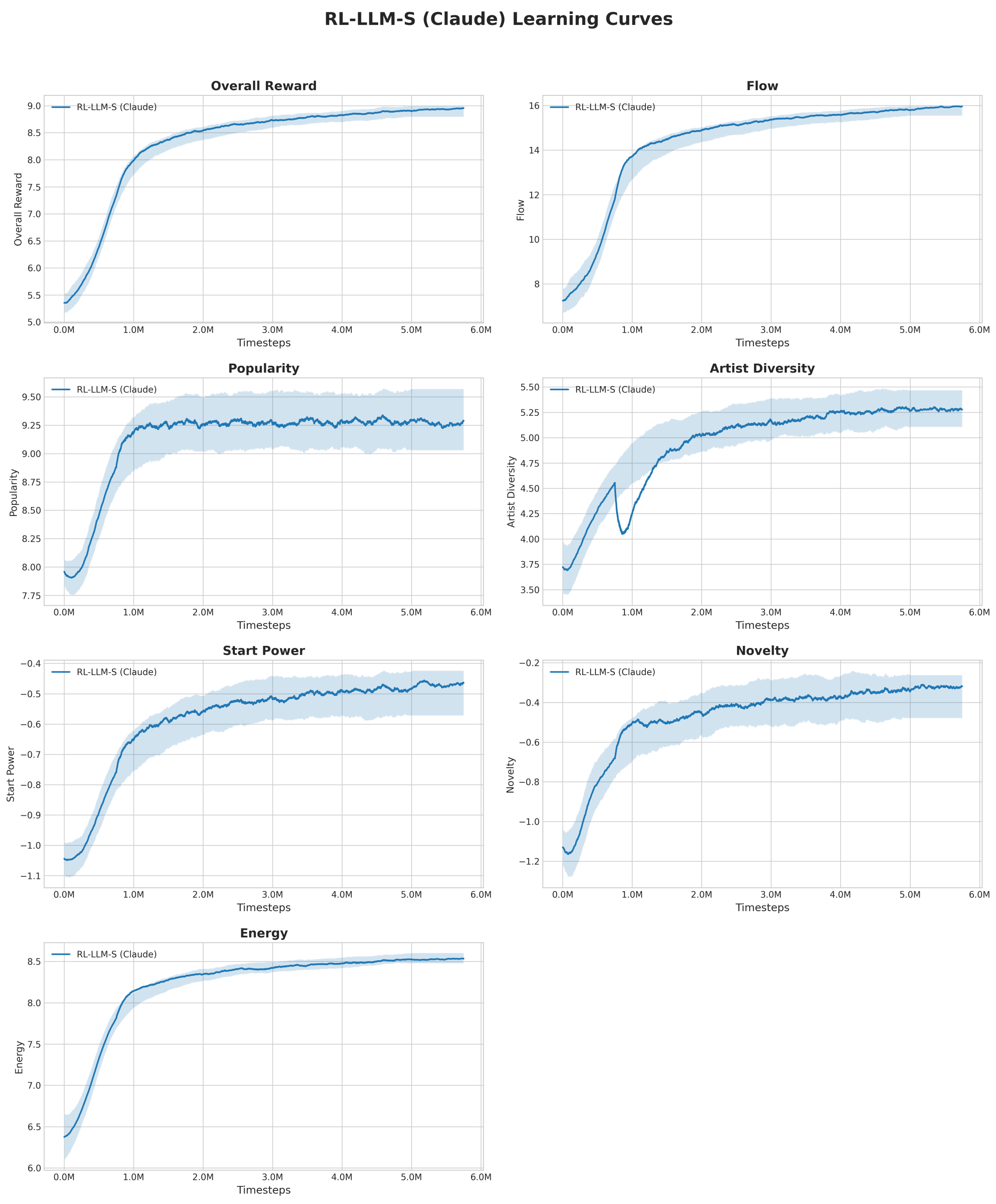


Figure D.4: Learning curves for the policy guided by rewards from **Claude 3 Opus (Summarized)**. The agent learns effectively, showing clear positive trends in key metrics like Novelty and Popularity, backed by a consistent, low-variance training profile.

D.3. Pipeline B: RL-LLM-R

The figures in this section show the learning curves for agents trained on reward functions generated from raw interview transcripts. Each figure corresponds to a policy trained using a different LLM.

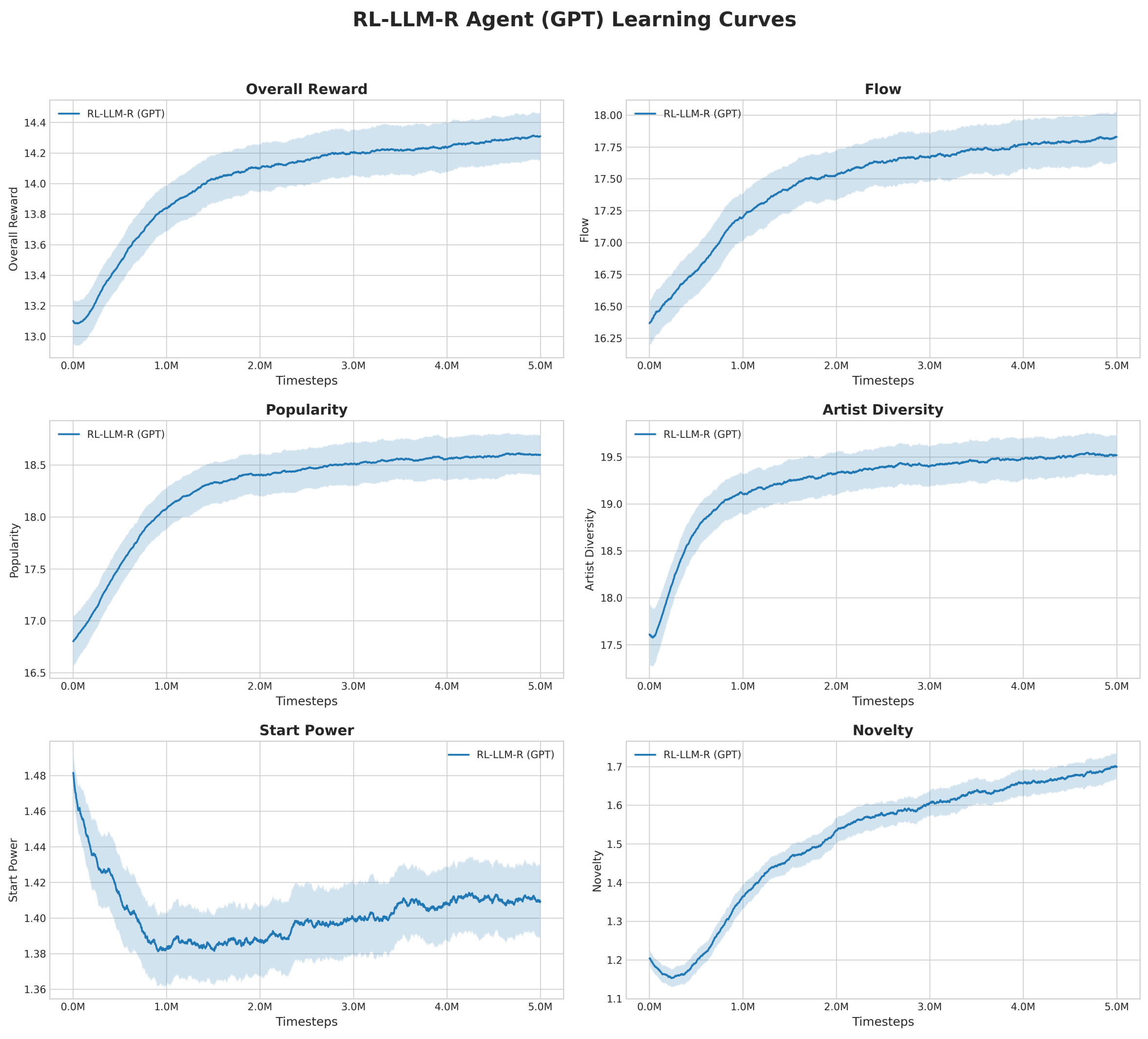


Figure D.5: Learning curves for the policy guided by rewards from **GPT-4o (Raw Transcripts)**. The agent exhibits a stable learning trajectory, suggesting that the model can effectively generate a coherent reward function even from noisy, unprocessed user feedback.

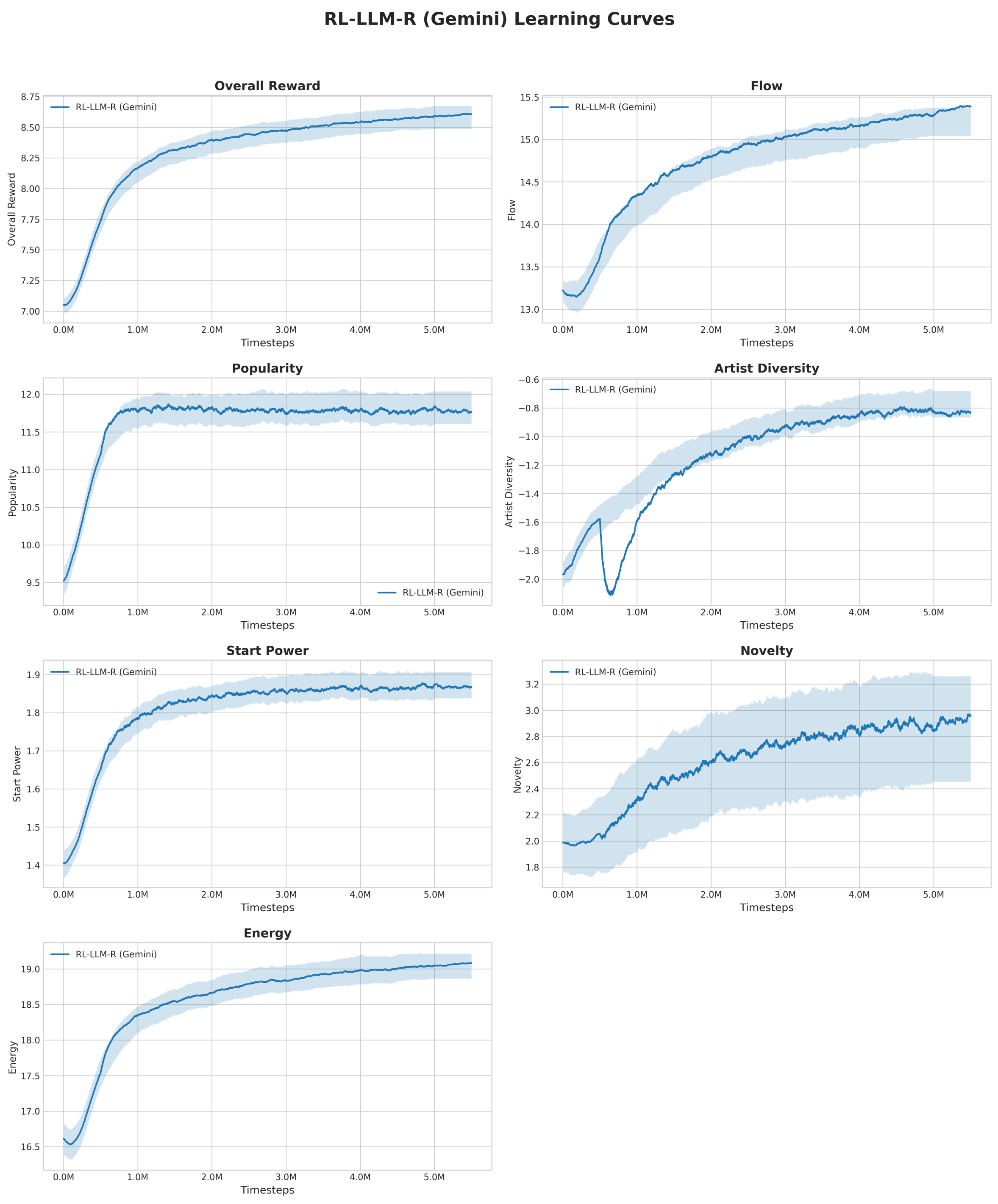


Figure D.6: Learning curves for the policy guided by rewards from **Gemini 2.5 Pro (Raw Transcripts)**. The tight standard deviation across the five runs highlights the reliability of the generated reward signal, leading to a predictable and stable training outcome.

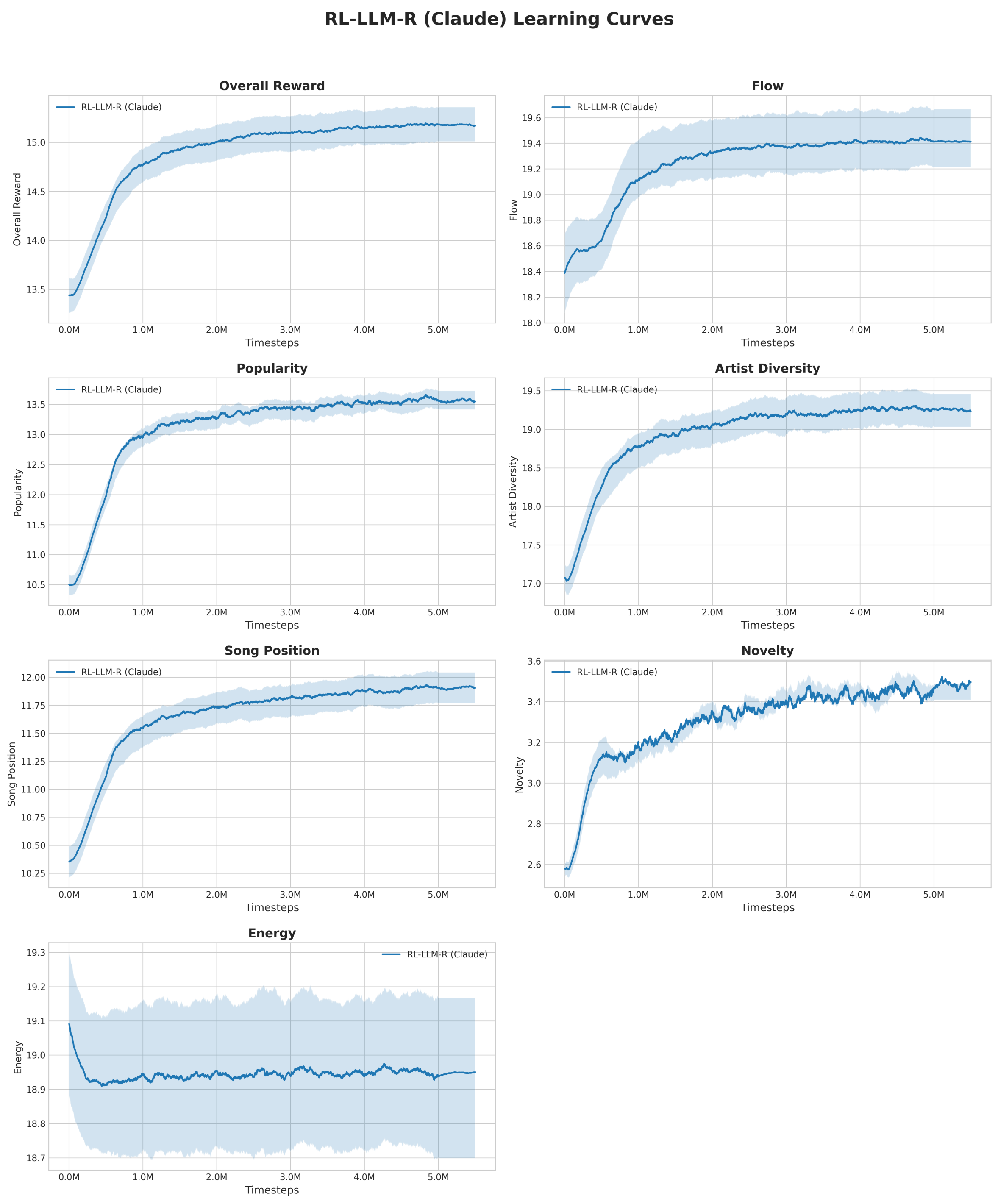


Figure D.7: Learning curves for the policy guided by rewards from **Claude 3 Opus (Raw Transcripts)**. The plots show successful and stable policy optimization, confirming that raw transcripts, when processed by a capable LLM, can serve as a robust source for reward generation.

D.4. Performance Stability Across Run

The following tables provide summary statistics for model performance, complementing the stability analysis in the main report. Each cell contains the mean (μ) and standard deviation (σ) calculated across five independent runs with different random seeds. The data in the tables confirm the stability of the models. Most LLM-guided agents exhibit very low variance across the five runs; for instance, the RL-LLM-R (Gemini) model's ndcg@k score is stable at 0.272 ± 0.004. The RL-M model serves as a slight outlier among the RL models, showing a comparatively higher standard deviation (e.g., 0.242 ± 0.013 for ndcg@k), indicating more variability in its performance. This overall consistency, among the RL agents, indicates that the aggregated results presented in the following sections are a reliable measure of each model's capabilities.

	RL-LLM-S			RL-LLM-R			RL-M
Metric	**Claude**	**GPT**	**Gemini**	**Claude**	**GPT**	**Gemini**	**Manual**
ndcg@k	0.271 ± 0.009	0.203 ± 0.004	0.272 ± 0.005	0.257 ± 0.003	0.110 ± 0.009	0.272 ± 0.004	0.242 ± 0.013
precision@k	0.272 ± 0.008	0.202 ± 0.004	0.271 ± 0.004	0.258 ± 0.005	0.117 ± 0.011	0.271 ± 0.004	0.239 ± 0.015
recall@k	0.158 ± 0.005	0.117 ± 0.001	0.158 ± 0.002	0.149 ± 0.003	0.070 ± 0.007	0.158 ± 0.002	0.137 ± 0.011

Table D.1: Summary of Recommendation Quality in the seedless scenario. Each cell shows the mean score and standard deviation ($\mu \pm \sigma$) calculated across five independent runs. Higher scores indicate better performance.

	RL-LLM-S			RL-LLM-R			RL-M
Metric	**Claude**	**GPT**	**Gemini**	**Claude**	**GPT**	**Gemini**	**Manual**
ndcg@k	0.268 ± 0.009	0.201 ± 0.004	0.269 ± 0.006	0.254 ± 0.003	0.108 ± 0.009	0.269 ± 0.004	0.239 ± 0.013
precision@k	0.267 ± 0.008	0.199 ± 0.004	0.267 ± 0.004	0.254 ± 0.005	0.114 ± 0.010	0.267 ± 0.005	0.235 ± 0.015
recall@k	0.154 ± 0.005	0.114 ± 0.001	0.153 ± 0.002	0.146 ± 0.003	0.067 ± 0.006	0.154 ± 0.003	0.133 ± 0.010

Table D.2: Summary of Recommendation Quality in the seeded scenario. Each cell shows the mean score and standard deviation ($\mu \pm \sigma$) calculated across five independent runs. Higher scores indicate better performance.

	RL-LLM-S			RL-LLM-R			RL-M
Metric	**Claude**	**GPT**	**Gemini**	**Claude**	**GPT**	**Gemini**	**Manual**
diversity	0.740 ± 0.009	0.720 ± 0.006	0.718 ± 0.017	0.711 ± 0.022	0.735 ± 0.002	0.711 ± 0.014	0.729 ± 0.004
flow	0.885 ± 0.003	0.767 ± 0.003	0.854 ± 0.005	0.835 ± 0.013	0.834 ± 0.004	0.863 ± 0.002	0.811 ± 0.014
novelty	0.128 ± 0.002	0.235 ± 0.001	0.162 ± 0.004	0.176 ± 0.012	0.182 ± 0.003	0.152 ± 0.002	0.199 ± 0.011
popularity	1.372 ± 0.009	1.436 ± 0.010	1.519 ± 0.016	1.525 ± 0.019	1.109 ± 0.018	1.517 ± 0.003	1.379 ± 0.053
start power	0.104 ± 0.005	0.118 ± 0.006	0.132 ± 0.007	0.129 ± 0.005	0.033 ± 0.004	0.127 ± 0.005	0.114 ± 0.017

Table D.3: Summary of Reward Components in the seedless scenario. Each cell shows the mean score and standard deviation ($\mu \pm \sigma$) calculated across five independent runs. These values indicate alignment with expert curatorial principles.

	RL-LLM-S			RL-LLM-R			RL-M
Metric	**Claude**	**GPT**	**Gemini**	**Claude**	**GPT**	**Gemini**	**Manual**
diversity	0.741 ± 0.009	0.722 ± 0.006	0.720 ± 0.017	0.713 ± 0.020	0.736 ± 0.001	0.713 ± 0.013	0.731 ± 0.004
flow	0.885 ± 0.003	0.766 ± 0.002	0.854 ± 0.005	0.835 ± 0.013	0.834 ± 0.004	0.863 ± 0.001	0.811 ± 0.013
novelty	0.128 ± 0.002	0.236 ± 0.001	0.162 ± 0.004	0.177 ± 0.012	0.183 ± 0.003	0.152 ± 0.002	0.199 ± 0.011
popularity	1.372 ± 0.011	1.438 ± 0.012	1.520 ± 0.017	1.527 ± 0.019	1.107 ± 0.020	1.518 ± 0.003	1.379 ± 0.052
start power	0.109 ± 0.005	0.124 ± 0.006	0.139 ± 0.007	0.135 ± 0.006	0.035 ± 0.005	0.133 ± 0.004	0.119 ± 0.018

Table D.4: Summary of Reward Components in the seeded scenario. Each cell shows the mean score and standard deviation ($\mu \pm \sigma$) calculated across five independent runs. These values indicate alignment with expert curatorial principles.