Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3705328.3748053

RESEARCH-ARTICLE

A Language Model-Based Playlist Generation Recommender System

ENZO CHAROLOIS-PASQUA, EURECOM, Sophia Antipolis, Provence Alpes Côte d'Azur, France

ELÉA VELLARD, EURECOM, Sophia Antipolis, Provence Alpes Côte d'Azur, France

YOUSSRA REBBOUD, EURECOM, Sophia Antipolis, Provence Alpes Côte d'Azur, France

PASQUALE LISENA, EURECOM, Sophia Antipolis, Provence Alpes Côte d'Azur, France

RAPHAËL TRONCY, EURECOM, Sophia Antipolis, Provence Alpes Côte d'Azur, France

Open Access Support provided by:

EURECOM

PDF Download
3705328.3748053.pdf
05 March 2026
Total Citations: 1
Total Downloads: 4739

Published: 07 September 2025

Citation in BibTeX format

RecSys '25: Nineteenth ACM Conference on Recommender Systems
September 22 - 26, 2025
Prague, Czech Republic

Conference Sponsors:
SIGCHI

A Language Model-Based Playlist Generation Recommender System

Enzo Charolois-Pasqua*
EURECOM
Sophia Antipolis, France
enzo.charolois-pasqua@eurecom.fr

Eléa Vellard*
EURECOM
Sophia Antipolis, France
elea.vellard@eurecom.fr

Youssra Rebboud
EURECOM
Sophia Antipolis, France
Youssra.Rebboud@eurecom.fr

Pasquale Lisena
EURECOM
Sophia Antipolis, France
pasquale.lisena@eurecom.fr

Raphaël Troncy
EURECOM
Sophia Antipolis, France
raphael.troncy@eurecom.fr

Abstract

The title of a playlist often reflects an intended mood or theme, allowing creators to easily locate their content and enabling other users to discover music that matches specific situations and needs. This work presents a novel approach to playlist generation using language models to leverage the thematic coherence between a playlist title and its tracks. Our method consists in creating semantic clusters from text embeddings, followed by fine-tuning a transformer model on these thematic clusters. Playlists are then generated considering the cosine similarity scores between known and unknown titles and applying a voting mechanism. Performance evaluation, combining quantitative and qualitative metrics, demonstrates that using the playlist title as a seed provides useful recommendations, even in a zero-shot scenario.

CCS Concepts

• **Information systems** → **Language models**; **Recommender systems**.

Keywords

Language model, Playlist generation, NLP

ACM Reference Format:
Enzo Charolois-Pasqua, Eléa Vellard, Youssra Rebboud, Pasquale Lisena, and Raphaël Troncy. 2025. A Language Model-Based Playlist Generation Recommender System. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3705328.3748053

1 Introduction

In the era of personalized digital experiences, recommender systems have become crucial for delivering content tailored to individual tastes. These systems use advanced algorithms and data analytics to understand user preferences and behaviors, making each interaction uniquely relevant. Music platforms, in particular, rely heavily on recommender systems to enhance user engagement and satisfaction by suggesting songs and artists that align with a user's listening habits and favorite genres.

However, creating tailored playlists remains a challenging and time-consuming task for users. Crafting a playlist that captures the right mood, includes a diverse yet cohesive selection of tracks, and adapts to different listening contexts requires significant effort. Users must manually search for songs, consider the sequence and flow of the playlist, and continuously update it to keep it fresh and engaging. This process can be overwhelming, especially for those with large music libraries or eclectic tastes.

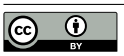
Apart from the tracks, another distinctive characteristic of a playlist is its title. Previous studies have demonstrated that the playlist titles significantly influence users' expectations, playing a crucial role in their selection process [17]. Moreover, playlists, and even private ones, have generally meaningful titles that reflect their content, mood, genre, or the context in which they were created, such as a specific situation, event, or purpose. As a result, it is common to find similarities in the titles of playlists created by different users, who often include words like "workout", "gaming", or "wedding" to indicate the playlist's purpose [14]. For these reasons, the playlist title can be a key element in recommender systems, specifically if the title reveals a commonsense theme rather than being personal [28]. First, it can serve as the sole seed element to suggest songs for an empty playlist in a *cold start* scenario. Secondly, it can enhance other types of recommender systems (e.g., collaborative-based) by adding a semantic constraint for proposing subsequent songs.

While it is true that titles like those mentioned in the previous paragraph may have abundant ready-made materials, e.g. by streaming services, a user may instead want to have a playlist named "Housewarming Party", "Spring awakening", or "Country summer"[1] will hardly have a straightforward, one-to-one match with existing categories.

In this paper, we specifically study this challenge in proposing a pipeline for playlist generation using a sole title as input. Unlike previous works [10, 19], we use language models to capture the semantic meaning of playlist and track titles. In particular, we fine-tune a pre-trained transformer-based language model on playlist groups, clustered based on their track content. Such a fine-tune

*Both authors contributed equally to this research.

[1]Those real playlist titles that one can find in the Million Playlists Dataset.



This work is licensed under a Creative Commons Attribution 4.0 International License.
RecSys '25, Prague, Czech Republic
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1364-4/25/09
https://doi.org/10.1145/3705328.3748053

language model enables the creation of vector representations for any playlist title, whether known or unknown, of any complexity and not necessarily matching a pre-existing category (genre, mood, emotion, purpose) as in previous related work [5, 11]. Track recommendations for a given playlist title are subsequently made using similarity metrics and a voting mechanism. In addition, we assess the capacity of Large Language Models (LLM) in playlist generation from titles, both in a 0-shot scenario and for ranking a pre-selection of tracks.

The main contributions of this work are:

- A pipeline for generating playlists from their titles;
- An evaluation of this pipeline, against different variants and against other title-based recommender systems;
- An assessment of the capabilities of prompt-based LLM in the task of playlist generation.

The code and data of all our experiments are available in open source at https://github.com/elea-vellard/LLM-Playlist-Recommender.

The remainder of this paper is structured as follows. Section 2 provides an overview of previous work related to playlist generation. Section 3 delves into the details of our proposed approach, outlining each step comprehensively. Section 4 presents and discusses the results of our study. Finally, Section 5 concludes the paper, highlights its limitations and outlines potential future work.

2 Related work

The inclusion of playlist titles in music recommender system research is relatively recent. Several experiments have been made in generating the title of a playlist, given the list of tracks it includes, using RNN and transformer-based models [8, 15]. In 2024, an online web application uses a combination of GPT-4 [22] and Claude [1] to generate good and catchy playlist titles given a list of songs.[2] While these studies investigate a task that is the inverse of ours, they provide some evidence that language models can successfully capture the semantic content of playlist titles.

An in-depth study on the creation and purpose of playlists was conducted in [5]. The authors of this study identified 9 distinct categories (excluding 'Other') that guide the organization of tracks into playlists. Furthermore, the research delves into examples of lengthy and complex playlist titles, which pose challenges when attempting to assign them to specific categories. In [11], a topic model applied on titles is used to classify playlists, and represent them as 10-dimensional topic mixture vectors. The distance between these vectors is used to recommend similar playlists. McFee and Lanckriet [18] apply natural language processing techniques to playlists by treating songs as tokens in a sequence, analogous to words in a sentence. While they do not use natural language directly, songs are modeled as discrete items similar to words in traditional NLP tasks. For example, a playlist containing the songs *Let It Be*, *Hey Jude*, and *Imagine* would be represented as a three-token sequence, just like a sentence composed of three words. Other works intersecting music information retrieval and NLP are reported in [13].

In 2018, the annual RecSys Challenge focused on playlist completion, leading to the release of the Million Playlist Dataset by Spotify [4]. This dataset remains one of the most significant resources for the task today. The inclusion of 0-seeds playlist in the challenge, pushed several participants in having a *cold start* strategy based on playlist titles. The underlying rationale for the proposed implementations is consistent across all participants: playlists with similar titles are more likely to contain the same tracks. In [10], titles are used to create a matrix reporting if a given track (column) appears in at least one playlist having a specific title (row); the matrix is then used to compute a playlist-song pair score. In [19], playlist embeddings have been computed based on the co-occurrences of tracks, and these embeddings are used for clustering and for training a *fastText* model [2], working at the n-gram level.[3] In [16], another approach relying on a LSTM that learns word embeddings (based on n-grams), used in input – together with track features – to a Recurrent Neural Collaborative Filtering model is proposed, showing good performances in the no-seed scenario.

Other textual information are used in the literature to generate or to continue playlists. In [12], a Knowledge Graph is used to connect the contextual information of a playlist – extracted from its metadata such as the title – and the genres of the tracks, which is then fed in a classifier. These contextual information are however manually annotated. Emotions are leveraged in [20], extracted with a LSTM from chatbot interactions and mapped to acoustic features of tracks that are proposed as a playlist. In this context, a sad emotion can lead to a selection of sad and slow songs. Text2Playlist [6] is a tool designed to generate relevant playlists based on a textual prompt (e.g., "I want music from the 90s for work"). The system uses an LLM to extract a set of tags from the prompt, which are then used to produce playlist recommendations which are further refined by the LLM. Finally, Text2Tracks [23] and TALKPLAY [7] perform a supervised finetuning of LLMs on pairs including a query (or conversation) and a list of recommended tracks.

These studies do not directly engage with titles, thus distinguishing from our work. Indeed, following the RecSys Challenge 2018, limited attention has been devoted to playlist titles, leaving the state of the art largely globally unchanged while transformer-based approaches and LLM enable nowadays to go one step further.

3 Method

Our approach to playlist generation exploits the textual information of playlist (name and track titles) to capture the intended scope and topic of the playlist itself. Our methodology is divided into two phases.

The **training phase** (Figure 1) begins with the preprocessing of the dataset to extract the relevant information for our goal. Afterwards, we employ a pre-trained transformer-based model to generate embeddings that capture the semantic meaning of the titles of both playlist and the included tracks. These embeddings transform the textual information into high-dimensional vectors, which are the representation of the playlists in a semantic space.

We apply a clustering algorithm to the generated embeddings, in order to group playlists based on their thematic similarity and identify distinct themes or moods that are prevalent across the dataset. The resulting clusters serve as the foundation for fine-tuning a language model, making it more capable to capture semantic similarity in this specific context.

[2] https://www.playlistnameai.com/

[3] An *n*-gram is a contiguous sequence of *n* items (phonemes, syllables, letters, etc.) from a given sample of text or speech.

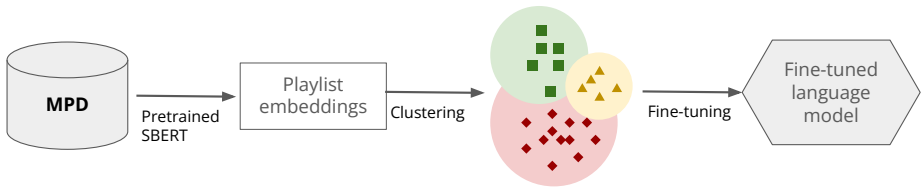


Figure 1: Diagram representing our training approach

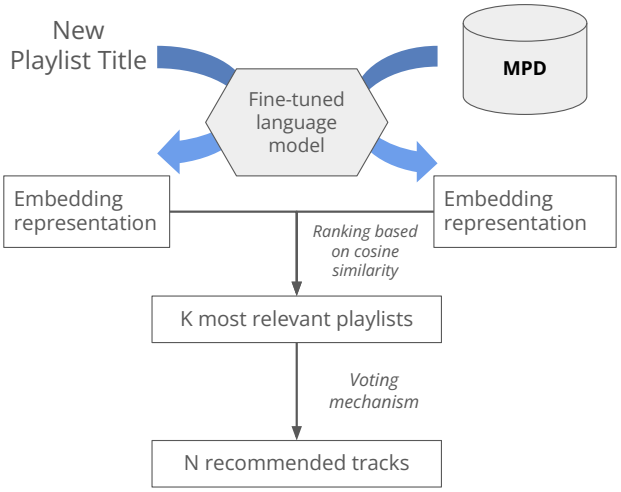


Figure 2: Diagram representing our recommendation approach when a new playlist title is used as seed

In the **generating phase** (Figure 2), the fine-tuned transformer model extracts the embedding representations of titles from both existing playlists and new input playlists. These embedding representations are then compared using cosine similarity scores. Each new input playlist title is compared with the known titles of existing playlists within the dataset, ranking them based on similarity. This process identifies the most similar playlists.

A voting mechanism is subsequently employed to determine the final selection of tracks in the new playlist being generated. This mechanism considers all tracks included in the most similar playlists, and returns in output those that occur the most.

Separately, we conducted additional experiments with prompt-based LLM for playlist generation from a title, in order to have a comparison with our proposed fine-tuned model.

These steps are further detailed in the following sections.

3.1 Processing the dataset

The Million Playlist Dataset (MPD) released by Spotify in 2018 [4] is composed of one million user-generated playlists that are commonly used for understanding music preferences and for developing recommendation algorithms. It is available as a set of JSON slices, each slice containing 1,000 playlists. In the dataset, a playlist is represented with a title and a unique playlist identifier (pid), and includes some additional information such as the timestamp of the last update, the number of included tracks and artist, and an (optional) description. Each playlist is composed of a certain number of tracks and artists, that can be identified by their unique Uniform Resource Identifier (URI).

To better manage the content, the dataset is transformed into easily-readable CSV files:

- *playlists.csv* contains the main information of each playlist: its pid, its title, the number of tracks, etc.
- *items.csv* serves as a bridge between tracks and their corresponding playlists. It connects every track (identified by its URI) to its playlist (pid) and writes the track's position within the playlist.
- *tracks.csv* contains information about each track: its unique URI, the title, the artist, the album URI and name, and the track duration.

3.2 Clustering strategy

We hypothesize that playlist titles and track names convey specific ideas and themes that maintain coherence within the context of the playlist, with this coherence being domain-specific rather than universal. For example, a playlist titled "Let it Snow" could simply suggest a winter theme when just considering the semantic meaning of the words. However, within the context of music playlists, it unmistakably indicates a Christmas-themed collection.

Based on this premise, we propose that playlist titles and track titles that frequently co-occur are more likely to share semantic relatedness, similar to how words that often appear together in sentences do. Just as sentences, when grouped into documents, provide the contextual foundation for fine-tuning language models, the combination of playlist and track titles, organized into thematically coherent clusters, can serve as a valuable dataset for fine-tuning a playlist-aware recommendation model.

3.2.1 Vector representation of playlist content. There are several methods to represent playlists as vectors based on their content. The one-hot encoding approach, as in [10], was deemed impractical due to its excessive size and computational inefficiency. The skip-gram approach, as described in [19], was also considered and discarded for similar reasons. Consequently, we opted to use text-based embeddings for this vector representation because they offer a more compact dimensionality through aggregation techniques. In this scenario, each playlist is represented by the titles of all its tracks.

Initially, we tested word embedding models such as *fastText* [2], averaging the vectors of all words composing the track titles in the playlist. However, our initial assessment revealed that this model was not sufficiently efficient for our use case, as it primarily focuses

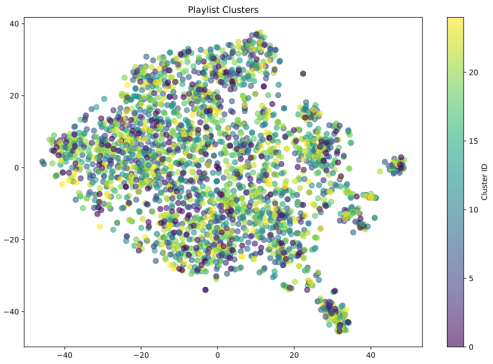


Figure 3: T-SNE plot of 10 clusters for the training set using kMeans on fastText embeddings

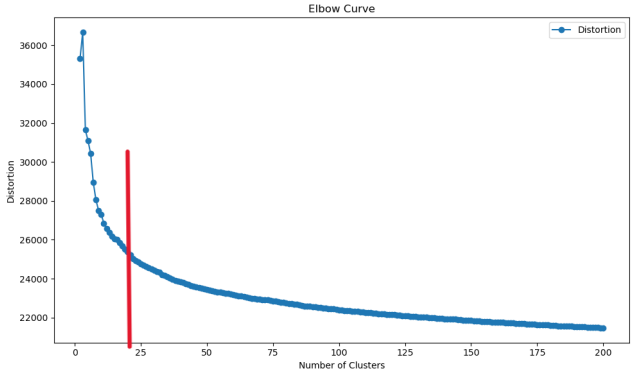


Figure 4: The elbow curve and the suggested number of clusters (in red)

on word-level representations and failed to capture the contextual meaning of playlist titles. This was particularly evident in our attempt to cluster playlists, which resulted in poorly defined data separations as illustrated in Figure 3.

Thus, we opted for a transformer-based model, which has demonstrated strong performance across a wide range of tasks [29]. In particular, we employed *Sentence BERT*, which leverages the attention mechanism through Siamese BERT-Networks. This approach involves two BERT blocks processing sentence pairs concurrently, followed by a pooling layer that generates fixed-size sentence embeddings [24]. This enables Sentence BERT to create more effective representations at the sentence-level. Given that playlist titles can be viewed as concise sentences (or at least are more extensive than single words), we conclude that Sentence BERT is the most appropriate choice for our task.

For each playlist, we compute the mean embedding e of all the track titles in the playlist itself:

$$e_{playlist} = \frac{1}{n} \sum_{i=1}^{n} e_{track_i}$$

where n is the number of tracks within the playlist, and e_{track_i} is the Sentence BERT embedding of the i-th track in the playlist.

By focusing solely on the tracks within the playlists (i.e., their content) and excluding the titles, we achieve two key objectives: (1) We prevent the introduction of bias that could arise from similar titles representing in facts different themes or concepts. (2) By concealing the titles during the clustering stage, we can confidently extract the test set later, ensuring it contains only titles that were not seen during training. This approach maintains the integrity of our evaluation process.

3.2.2 Clustering algorithm. The playlist vectors are clustered using the K-Means algorithm. We manually selected the number of clusters instead of relying on density-based algorithms like DBSCAN. This was because the embedding space, composed of mean embeddings, was very dense, making it difficult for DBSCAN to effectively differentiate between various themes. For the same reason, metrics such as the Silhouette or Davies-Bouldin scores, which measure intra- and inter-cluster distances, were not particularly relevant in our specific use case.

We implemented an elbow curve approach to estimate an optimal number of clusters, as in Figure 4. Since our main goal was to ensure coherence within each cluster, the elbow curve served more as a guideline to establish a minimum value rather than a definitive choice for the number of clusters. In fact, although the elbow curve suggested a number k of clusters around 25, we ultimately set $k = 200$. We observed that increasing the number of clusters generally improved performance, as visible in Figure 5. However, we also found that a higher number of clusters significantly increased computational cost. Beyond 200 clusters, the computational expense became prohibitive. Therefore, based on our analysis, we settled on 200 clusters as the optimal choice.

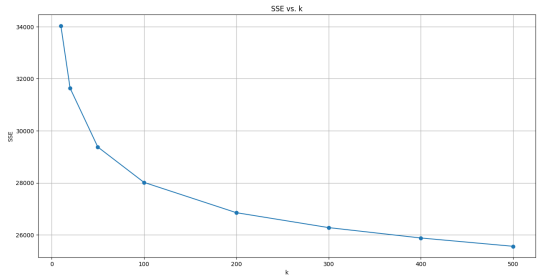


Figure 5: The standard error of estimate (SSE) of clusters decreases with the increment of number k of clusters.

The T-SNE plot of an initial experiment made with 50 clusters (Figure 6) reveals two key observations. First, the clusters are more distinct compared to those in Figure 3, although some overlaps persist, making them challenging to separate completely. Additionally, the presence of clusters that are significantly larger and more dispersed than others is evident. This phenomenon can be attributed

to certain clusters containing a broader range of topics[4], which we will refer to as *miscellaneous* throughout this paper.

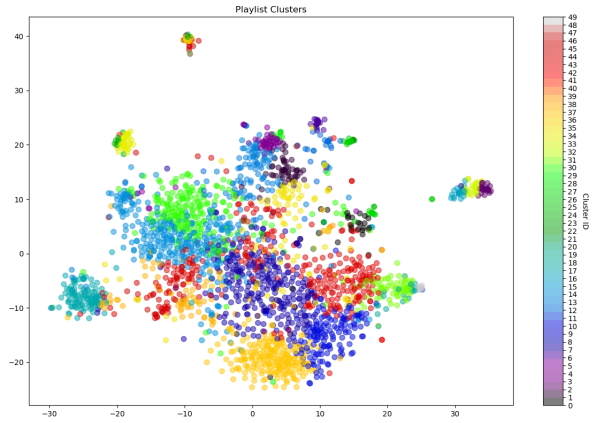


Figure 6: T-SNE plot of 50 clusters for the training set using kMeans on Sentence BERT embeddings. It is possible to note at least two *miscellaneous* clusters, in dark blue and green.

The presence of these *miscellaneous* clusters poses several challenges. First, their significantly larger size compared to other clusters can lead to imbalanced training, potentially impacting classification accuracy. Secondly, the informative value of these clusters is questionable, as their playlists are included primarily because they do not fit well into other categories. Therefore, we decided to remove those clusters (and playlists) from our training set. Indeed, for the goal of fine-tuning a language model, we prefer having coherent clusters (including only playlists on the same theme), rather than complete ones (with a unique cluster theme). In order to realize this, we found for each cluster the most occurring playlist title and we computed the percentage of exact matches with it among all the playlists' titles of that cluster. We consider valid the clusters that have the percentage over a threshold t and miscellaneous below it. The threshold has been set to $t = 0.02$ empirically. Indeed, most clusters below this percentage include a number of playlists largely above the mean of all other clusters. Although this method is relatively simple, we found it to be effective in identifying miscellaneous clusters, which have been excluded from the dataset. This action removed around 40% of the MPD, resulting in a total of 158 clusters left for the next sections[5].

We split our processed dataset into training (80% of the data), validation (10%) and test set (10%), ensuring a representation of each cluster in every set, so that the model can train and evaluate its performance on specific clusters.

3.3 Fine-tuning a language model

The training and validation set created in the previous step is used to fine-tune a language model, with the ultimate goal of effectively distinguishing and classifying playlist themes. The input for fine-tuning consists of clusters, each represented by concatenating the titles of its associated playlists. These concatenations of titles acts as documents in our fine-tuning, while cluster IDs are used as labels.

Concerning the architecture, we used a sentence transformer backbone [24] and in particular the `all-MiniLM-L6-v2`[6] model. This model maps sentences to a 384 dimensional dense vector space and can be used for tasks like clustering, while being compact enough to ensure computational efficiency. Additional classification layers were included, composed of a dropout layer for regularization and a fully connected layer to predict cluster labels. The model is fine-tuned in two variants:

- using the *cross-entropy loss*, commonly used for classification tasks;
- using the *triplet loss*, which focuses on making items from the same group closer than those from different groups [25].

We used standard training arguments, including a batch size of 8 and a learning rate of 2×10^{-5}. Throughout the process, we monitored the loss on both the training and validation sets to detect any signs of overfitting, which we observed after approximately 15 epochs. Consequently, we stopped the fine-tuning early and retained the model that minimized the validation loss.

On the output model, we observed significant improvements in the model's ability to adapt to the music playlist context. To illustrate this, we compared the cosine similarity scores between the embedding of the word "Workout" and each of the words "K-pop", "Running", and "Sports". In Figure 7, it is evident how the three model variants differ in the representation. The pre-trained

[4]In particular, the presence in the MPD of some broad-sense, not-meaningful playlist titles, such as "music" or "my playlist" was already reported in [10].
[5]It is important to specify that we exclude the miscellaneous clusters only during the fine-tuning phase. For this task, a more homogeneous corpus was required to prevent the language model from learning irrelevant associations. However, any title can subsequently be processed by the fine-tuned model, which also leverages its pre-existing knowledge.
[6]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

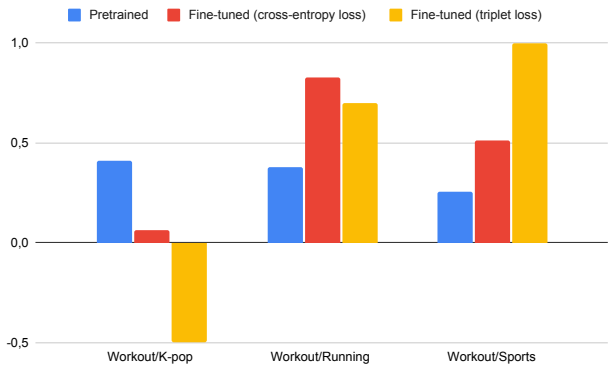


Figure 7: The cosine similarity scores of pairs of embeddings computed with the three model variants.

model exhibits less variability in its vectors, indicating a more uniform and compact embedding space where nodes are closer to each other. In contrast, fine-tuning helps to better differentiate semantically distant words, such as "Workout" and "K-pop", while bringing semantically similar words, like "Workout" and "Running" or "Workout" and "Sports", closer together. It is noteworthy that in the triplet loss variant, "Workout" is closer to "Sports" than to "Running". While "Workout" and "Sports" are definitively within the semantic realm of physical activity, "Running" can also appear in metaphorical contexts – such as "running away from sentiments" – and this has probably been captured in the fine-tuned model.

3.4 Playlist generation

The final step of our pipeline focuses on generating playlists that align with a user's input title. Utilizing the fine-tuned model, we aim to recommend tracks that match the input theme, even if they are no exact matches with the proposed title.

The embeddings of each (known) playlist titles in the dataset are computed using the finetuned model. As seed for the generation process, a (unknown) playlist title – that can be composed of one or multiple words – is provided, and the embedding of the title is generated. The latter is compared with the embeddings of all known playlists using cosine similarity.

Formally, the cosine similarity between two embeddings e_a and e_b is defined as:

$$\text{cosine_similarity}(\mathbf{e}_a, \mathbf{e}_b) = \frac{\mathbf{e}_a \cdot \mathbf{e}_b}{\|\mathbf{e}_a\| \|\mathbf{e}_b\|}$$

The 50 playlists with the highest similarity scores are selected. Among these, we count the occurrences of each track. A **voting mechanism** is applied, where tracks that appear more frequently across the matched playlists receive higher scores, indicating their relevance to the input theme. The top k most frequently occurring songs are then recommended to the user.

3.5 Playlist generation using LLM

We also conducted a playlist generation experiment using three Large Language Models in a implementation using LangChain [3]. In particular, we tested a zero-shot and two few-shot scenarios, including 1 and 5 songs respectively as examples given in the prompt. To ensure a fair comparison with our ground truth, we explicitly instructed the LLM to consider only songs published before the release date of the MPD dataset.

The prompt provided to the LLM is structured as follows. We first assign the LLM the role of an expert in music recommendation and clearly define its task. We then explicitly outline a set of constraints that the LLM must adhere to, restricting its recommendations to songs released before 2017. We ask to generate a playlist of 10 songs, and specify that the output must exclusively be in JSON format to simplify subsequent parsing. Elements in curly brackets ({}) represent configurable parameters. This allowed us to obtain a simple format of recommended songs in order to easily compute quantitative metrics.

Prompt for LLM Playlist Generation

You are an expert in music playlist generation.
Your task is to generate the continuation of a playlist given only its title and five example songs with their artists.
Important:

- You have to select only songs released before October 2017.
- Propose a COMPLETE playlist consisting of exactly 10 songs.
- Do not output any extra text besides the list of songs.
- The output must be a JSON list where each item is a dictionary with keys `"song"` and `"artist"`.
- Only return the JSON list. Do not repeat the instructions or inputs.
- In each dictionary, the value for `"song"` must contain only the song name (without any hyphen or artist information) and the value for `"artist"` must contain only the artist name.
- All recommended songs must be UNIQUE and must not repeat any of the five example songs provided.

Playlist Title: `"{playlist_title}"`
Examples:
(1) `{"song": "{song1}", "artist": "{artist1}"}`
(2) ...
Output format (strict):

```
[
  {"song": "<title>", "artist": "<artist>"},
  ...
]
```

Answer ONLY with the JSON list exactly as specified above. Do not output anything else.

4 Evaluation

To evaluate the effectiveness of our method, we employed a combination of quantitative and qualitative metrics. The quantitative evaluation focuses on commonly used metrics, computed on the playlists from the test set. It is important to recall that the playlists in the test set have not been used during training.

For each playlist, the set of relevant songs R is defined as the actual tracks contained within that playlist. The set of recommended songs is noted as S. The metrics we focus on are computed for a playlist of N songs [4, 26]:

- **precision@N** measures the proportion of recommended tracks that are relevant, with N being the number of recommended songs.

$$precision@N = \frac{len(R \cap S)}{len(S)} \tag{1}$$

- **recall@N** assesses the proportion of relevant tracks that were successfully recommended.

$$recall@N = \frac{len(R \cap S)}{len(R)} \tag{2}$$

- **MRR@N** (Mean Reciprocal Rank) estimates the rank of the first relevant song in the recommended songs.

$$MRR@N = \frac{1}{Rank\ of\ the\ first\ relevant\ song\ in\ S} \tag{3}$$

- **R-Precision** evaluates the proportion of recommended tracks S_T and artists S_A that are relevant among all known relevant tracks G_T and artists G_A.

$$\text{R-precision} = \frac{|S_T \cap G_T| + 0.25 \cdot |S_A \cap G_A|}{|G_T|} \tag{4}$$

- **NDCG** (Normalized Discounted Cumulative Gain) evaluates the ranking quality of recommended tracks, improving as relevant tracks are positioned higher in the list. If r_i is the position in the ground truth of the i-th recommended track, over the N recommended songs, and $IDCG$ is the DCG obtainable when the recommended playlist matches perfectly the ground truth:

$$\text{DCG} = \sum_{i=1}^{N} \frac{r_i}{\log_2(i+1)} \tag{5}$$

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \tag{6}$$

To complement these quantitative metrics, we incorporated a qualitative evaluation process involving human reviews. This step assessed the thematic coherence and overall appeal of the generated playlists. Two reviewers, leveraging their familiarity with the songs and conducting listening sessions when necessary, evaluated each track within the playlists and assigned an overall qualitative score. The integration of qualitative evaluation is essential because quantitative metrics may not fully reflect the model's quality, given that the recommended tracks may still be relevant according to the playlist title, even if not among the involved ground truth.

4.1 Quantitative results

We evaluated each approach using the same metrics on the playlists from the test set. The two fine-tuned models – with cross-entropy loss and with triplet loss – are compared with a pre-trained Sentence BERT. For each model, we computed metrics on the first 10, 66, and 500 tracks for input playlist title, where 66 is the average number of tracks per playlist across the whole dataset, and 500 is the number of requested track to predict in RecSys Challenge 2018.

Table 1 shows the average metrics computed over multiple input playlists. We observed a positive impact from fine-tuning, with all metrics showing approximately double the improvement compared to the pre-trained model. The results between the two fine-tuned model variants are similar, although the cross-entropy variant performed slightly better across almost all metrics. When predicting 500 tracks, the fine-tuning with cross-entropy still yields better scores, although the improvement over the pre-trained model is less pronounced. Conversely, the triplet loss approach results in comparatively lower performance. As the number of predicted songs increases, popular songs are more frequently selected because they are prevalent across various playlists. This prevalence can diminish the distinctions between different variants, and this is reflected in the results.

Metric	PT	FT-C	FT-T
Precision@10	0.1630	**0.1793**	0.1616
Recall@10	0.0332	**0.0382**	0.0316
MRR@10	0.2567	**0.3254**	0.3004
R-Precision@10	0.0358	**0.0496**	0.0417
NDCG@10	0.2780	**0.3740**	0.3472
Precision@66	0.0571	**0.1228**	0.1176
Recall@66	0.0647	0.1383	**0.1424**
MRR@66	0.1896	**0.3542**	0.3444
R-Precision@66	0.0477	**0.1332**	0.1059
NDCG@66	0.2742	**0.4311**	0.4156
HIT@500	0.3868	**0.3873**	0.3329
Precision@500	0.0480	**0.0489**	0.0410
Recall@500	0.3868	**0.3979**	0.3329
MRR@500	**0.3499**	0.3490	0.2893
R-Precision@500	**0.1570**	0.1556	0.1285
NDCG@500	0.2731	**0.2825**	0.2297

Table 1: Quantitative scores of our method in the three studied variants: the pre-trained model (PT), the fine-tuned model with cross entropy loss (FT-C), and the fine-tuned model with triplet loss (FT-T)

Method	R-Precision	NDCG
Monti et. al [19]		
- *Only title*	0.0837	0.1260
- *Best*	*0.1634*	*0.1717*
Faggioli et. al [10]		
- *Only title*	0.1093	0.2451
- *Best*	*0.2078*	*0.3713*
Kim et. al [16]		
- *Only title*	0.0760	0.1866
- *Best*	*0.1924*	*0.3394*
Our method		
Pre-trained	**0.1570**	0.2731
Fine-tuned (cross)	0.1556	**0.2825**
Fine-tuned (triplet)	0.1285	0.2297

Table 2: Comparison of our method and other playlist generation solutions relying on the playlist title from RecSys challenge 2018

In Table 2, we compare our strategy with those in previous works. The figures for the methods in [10, 16, 19] are sourced from their respective papers. We include results computed using only title information, specifically R-Precision@500 and NDCG@500. Additionally, we present the best results from the relative papers, which also incorporate training over tracks and are marked as *Best* in the table. These *Best* results are not directly comparable with our approach since they have been obtained using track content while our approach relies solely on the playlist title.

Our method demonstrates superior results compared to previous works, in both reported metrics, surpassing of several points the performance of the highest competitors that use only the title information. Our method outperforms in NDCG also some of the

best variants, that incorporated additional information about the tracks in the training.

4.2 Qualitative assessment

In our qualitative evaluation, we conducted a human assessment of thematic coherence and appeal for the generated playlists. Due to the high costs of human annotation, we limited our evaluation to a set of 22 specifically chosen playlists from the test set, reported in Table 3, selected according to the following criteria:

- the playlist should include at least 10 tracks;
- a clear thematic alignment between playlist titles and their tracks should be present. In other words, we excluded titles that were too generic or not enough representative of the playlist content;
- both niche and more broadly themed playlists should be included in the final set.

PID	playlist title	nb. tracks
673925	KPOP	22
677580	workout music	63
321143	Dance	37
923247	Rock	198
301195	Summer	37
490485	Hawaii	36
575612	Classic Country	174
269088	older songs	8
606436	2016	129
701866	Dance	23
608829	FINESSE	55
273344	Oldies	239
501054	Rock	73
750528	sports	72
684261	Christian	10
44648	gaming	33
837665	classics	52
786219	Party	74
47214	workout	99
889395	work	51
497427	Love songs	43
677006	Summer	29

Table 3: List of 22 playlists used in our evaluation, with the number of tracks included in each of them

Each recommended track for a given playlist title was assessed by two reviewers, who reached a consensus on whether to consider each track as valid. All decisions were thus agreed by consensus by both reviewers. For instance, for a hypothetical playlist titled "Rock classics", tracks such as "Highway to Hell" by AC/DC, "Smells Like Teen Spirit" by Nirvana, and "It's My Life" by Bon Jovi would be considered highly relevant, while "My Heart Will Go On" by Céline Dion have to be considered as not fitting. The assigned qualitative score is the percentage of valid tracks over the total, computed as:

$$Qualitative\ Score = \frac{nb.\ valid\ tracks}{total\ reccomended\ tracks} \tag{7}$$

The metric is in the [0,1] range, being 1 when all recommended tracks are valid.

The results in Table 4 demonstrate that, while quantitative metrics offer valuable insights, qualitative assessments provide crucial additional perspectives. Specifically, although quantitative scores may not perfectly align with user-generated playlists, the good qualitative metrics across all three models underscore their ability to create meaningful and coherent playlists, with over 70% of tracks being relevant. The qualitative score confirms the superiority of the model fine-tuned with cross-validation.

Metric	PT	FT-C	FT-T
Qualitative Score@10	0.7376	**0.7789**	0.7533
Qualitative Score@66	0.7231	**0.7719**	0.7461

Table 4: Qualitative scores of our method in the three studied variants: the pre-trained model (PT), the fine-tuned model with cross validation loss (FT-C), and the fine-tuned model with triplet loss (FT-T).

4.3 Evaluation of the LLM generation

We conducted an initial assessment of how LLM perform using the same 22 playlists that were used for the qualitative evaluation. We tested three different LLMs, namely Llama 3.1 [9], Zephyr[7] [27], and GPT4o [21] in order to have representatives among open weights models (Llama and derivatives) and closed models (GPT). The results are presented in Table 5. Although the quantitative metrics are based on a different test set than those in Table 1[8], this experiment can already give us some insight on the LLMs performance.

All language models benefit from including a few examples in the prompt. Llama has a unique behavior, as it prefers 1 to 5 examples for most metrics, though not for qualitative assessments. In every configuration, GPT4o demonstrates the best performance. In addition, we noticed that Llama and in particular Zephyr were not completely respecting the provided instruction, recommending songs published after the desired date specified in the prompt. GPT4o still falls short compared to the fine-tuned system, even in the 5-shot scenario. When comparing zero-shot performance – which relies solely on title information as in the fine-tuned system – the gap becomes quite pronounced. However, when examining the qualitative results, the gap is less pronounced (Table 4), with over 60% of the recommended tracks being of good quality. While Llama and Zephyr fail to match the scores of all transformer-based models, GPT4o succeeds in doing so, albeit only in the 5-shot scenario.

The limited scope of this experiment serves as a preliminary step to assess the feasibility of using LLMs in playlist generation. Despite its preliminary nature, we believe there are valuable outcomes from this exploration:

- LLMs struggle to replicate the retrieval scores achieved in our previous experiment;
- However, the relevance of the songs proposed by LLMs remains qualitatively similar.

[7] https://huggingface.co/TheBloke/zephyr-7B-beta-AWQ
[8] For reference, the pre-trained Sentence BERT had Precision 0.0586, Recall 0.1295, and MRR 0.2189 on the same test set of 22 playlists.

Model	Llama			Zephyr			GPT4o		
n-shot	**0**	**1**	**5**	**0**	**1**	**5**	**0**	**1**	**5**
Precision@10	0.0591	0.0950	0.0636	0.0227	0.0409	0.0590	***0.0636***	0.1091	**0.1227**
Recall@10	0.0067	0.0173	0.0059	0.0019	0.0060	0.0103	***0.0073***	0.0122	**0.0197**
MRR@10	0.0966	0.1883	0.1723	0.0795	0.1318	0.1800	***0.1636***	**0.2871**	0.2505
R-Precision@10	0.0137	0.0269	0.0104	0.0062	0.0103	0.0171	***0.0157***	0.0219	**0.0338**
NDCG@10	0.1199	0.2102	0.1863	0.0952	0.1436	0.2011	***0.1900***	0.3039	**0.3249**
Qualitative	0.6416	0.6818	0.7132	0.6777	0.6945	0.7038	***0.7175***	0.7724	**0.7953**

Table 5: Performance metrics for LLM. In bold, the best absolute scores, while in *italic* the best scores for a zero-shot scenario

The real advancement in this context would be to integrate LLMs with other retrieval techniques, as outlined in the conclusions.

5 Conclusion and Future Work

Our study investigated the use of language models for playlist generation. The experiments confirmed that the playlist title is a valuable source of information, and capturing its semantic meaning can be particularly beneficial in cold start scenarios. By combining clustering with fine-tuning, we enhanced the model's ability to generalize across diverse playlist themes, providing a unique and complementary approach to more traditional recommender system techniques. When relying solely on the title, our results outperform existing work in the literature and LLM generation in zero-shot and few-shot settings.

Our approach empowers users to transcend the limitations of predefined categories, having the full potential of their choice of words as seed for generation. This opens up virtually unlimited combinations, offering users a novel way to discover relevant and potentially new and unexpected music. However, this work presents several limitations that necessitate future research.

Firstly, the voting mechanism tends to flatten differences for very common playlist titles, Moreover it tends to favor popular tracks, particularly for generic titles. Our implementation is designed to maximize retrieval-based metrics, as it was the evaluation focus of the RecSys Challenge 2018, which allows us to make meaningful comparisons with the state-of-the-art. Introducing metrics to capture diversity, novelty, and order would undoubtedly enhance the outcomes. For example, for recommending novel songs not yet included in any playlist, it would be possible incorporate Sentence-BERT similarity between a novel song title and existing playlists' title. This approach may result in lower precision metrics, as it prioritizes novelty over exact matches; the right trade-off between the metric would depend on the use case. Additionally, two strategies, could immediately help mitigate the popularity bias:

- with a ranking system powered by language models, as demonstrated in [6]. The final ranking of tracks can indeed include more than the 50 most similar playlists, introducing more variability in the results;
- use our system for playlist continuation, where the first N songs are used as seeds, to better compare with other systems. Alternative language models can be explored as substitutes for Sentence BERT to provide a more comprehensive comparison of their performance and capabilities.

Furthermore, specializing the model for specific languages could enhance its performance and relevance in diverse linguistic contexts. Implementing the system in a production environment would also be crucial to assess its real-world effectiveness and robustness. This step would involve optimizing for scalability and efficiency, ensuring the model can handle large volumes of data and user interactions seamlessly. In addition, we plan to include explainability of the recommendation system by providing users with insights into why certain tracks were recommended. Lastly, we believe that the value of this work has to be tested in combination with a collaborative-based method, exploiting the best of the two approaches.

Acknowledgments

This work was supported by the French Public Investment Bank (Bpifrance) i-Demo program within the LettRAGraph project (Grant ID DOS0256163/00).

No AI tools were used for data analysis, experimentation, or the formulation of conclusions, except in the ways it is described in the paper itself.

Author Contributions

Enzo Charolois–Pasqua: Methodology, Software, Writing – review & editing; **Eléa Vellard:** Methodology, Software, Writing – review & editing; **Youssra Rebboud:** Supervision, Methodology, Writing – review & editing; **Pasquale Lisena:** Conceptualization, Supervision, Methodology, Writing – original draft; **Raphael Troncy:** Supervision, Writing – review & editing;

References

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073 [cs.CL] https://arxiv.org/abs/2212.08073

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[3] Harrison Chase. 2022. *LangChain.* LangChain AI. https://github.com/langchain-ai/langchain

[4] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. Recsys challenge 2018: automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada)

(RecSys '18). Association for Computing Machinery, New York, NY, USA, 527–528. doi:10.1145/3240323.3240342

[5] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 2006. 'More of an Art than a Science': Supporting the Creation of Playlists and Mixes. In *7th International Conference on Music Information Retrieval (ISMIR)*. Victoria, Canada, 240–245.

[6] Mathieu Delcluze, Antoine Khoury, Clémence Vast, Valerio Arnaudo, Léa Briand, Walid Bendada, and Thomas Bouabça. 2025. Text2Playlist: Generating Personalized Playlists from Text on Deezer. In *47th European Conference on Information Retrieval (ECIR). Industry Talk*. Lucca, Italy. https://arxiv.org/abs/2501.05894

[7] Seungheon Doh, Keunwoo Choi, and Juhan Nam. 2025. TALKPLAY: Multimodal Music Recommendation with Large Language Models. arXiv:2502.13713 [cs.IR] https://arxiv.org/abs/2502.13713

[8] Seungheon Doh, Junwon Lee, and Juhan Nam. 2021. Music Playlist Title Generation: A Machine-Translation Approach. In *Proceedings of the 2nd Workshop on NLP for Music and Spoken Audio (NLP4MusA)*, Sergio Oramas, Elena Epure, Luis Espinosa-Anke, Rosie Jones, Massimo Quadrana, Mohamed Sordo, and Kento Watanabe (Eds.). Association for Computational Linguistics, Online, 27–31. https://aclanthology.org/2021.nlp4musa-1.6/

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024). doi:10.48550/arXiv.2407.21783

[10] Guglielmo Faggioli, Mirko Polato, and Fabio Aiolli. 2018. Efficient Similarity Based Methods For The Playlist Continuation Task. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 15, 6 pages. doi:10.1145/3267471.3267486

[11] Benjamin Fields, Christophe Rhodes, and Mark d'Inverno. 2010. Using Song Social Tags and Topic Models to Describe and Compare Playlists, In WOMRAD 2010 Workshop on Music Recommendation and Discovery, colocated with ACM RecSys 2010. *CEUR Workshop Proceedings* 633.

[12] Giovanni Gabbolini and Derek Bridge. 2023. Predicting the Listening Contexts of Music Playlists Using Knowledge Graphs. In *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 330–345.

[13] Giovanni Gabbolini and Derek Bridge. 2024. Surveying More Than Two Decades of Music Information Retrieval Research on Playlists. *ACM Trans. Intell. Syst. Technol.* 15, 6, Article 114 (Nov. 2024), 68 pages. doi:10.1145/3688398

[14] Yun Hao. 2021. *Towards a better understanding of music playlist titles and descriptions.* Ph. D. Dissertation. University of Illinois at Urbana-Champaign.

[15] Haven Kim, Seungheon Doh, Junwon Lee, and Juhan Nam. 2023. Music Playlist Title Generation Using Artist Information. In *The AAAI-23 Workshop on Creative AI Across Modalities*. Open Rewier, Washington DC, USA. https://openreview.net/forum?id=nmtmjfJQLS

[16] Jaehun Kim, Minz Won, Cynthia C. S. Liem, and Alan Hanjalic. 2018. Towards Seed-Free Music Playlist Generation: Enhancing Collaborative Filtering with Playlist Title Information. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 14, 6 pages. doi:10.1145/3267471.3267485

[17] Zhi Li, Min Song, Shen Duan, and Zhe Wang. 2022. Are users attracted by playlist titles and covers? Understanding playlist selection behavior on a music streaming platform. *Journal of Innovation & Knowledge* 7, 3 (2022), 100212. doi:10.1016/j.jik.2022.100212

[18] Brian McFee and Gert R. G. Lanckriet. 2011. The Natural Language of Playlists. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, Anssi Klapuri and Colby Leider (Eds.). Miami, Florida, USA, 537–542. http://ismir2011.ismir.net/papers/PS4-11.pdf

[19] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. 2018. An Ensemble Approach of Recurrent Neural Networks using Pre-Trained Embeddings for Playlist Completion. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 13, 6 pages. doi:10.1145/3267471.3267484

[20] Amrita Nair, Smriti Pillai, Ganga S Nair, and Anjali T. 2021. Emotion Based Music Playlist Recommendation System using Interactive Chatbot. In *2021 6th International Conference on Communication and Electronics Systems (ICCES)*. IEEE, Coimbatore, India, 1767–1772. doi:10.1109/ICCES51350.2021.9489138

[21] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger,

Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[22] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[23] Enrico Palumbo, Gustavo Penha, Andreas Damianou, José Luis Redondo García, Timothy Christopher Heath, Alice Wang, Hugues Bouchard, and Mounia Lalmas. 2025. Text2Tracks: Prompt-based Music Recommendation via Generative Retrieval. *arXiv preprint arXiv:2503.24193* (2025).

[24] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, 815–823. doi:10.1109/CVPR.2015.7298682

[26] Guy Shani and Asela Gunawardana. 2011. *Evaluating Recommendation Systems*. Springer US, Boston, MA, 257–297. doi:10.1007/978-0-387-85820-3_8

[27] Lewis Tunstall, Edward Emanuel Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M Rush, and Thomas Wolf. 2024. Zephyr: Direct Distillation of LM Alignment. In *First Conference on Language Modeling*. OpenReview, Philadelphia, PA, USA. https://openreview.net/forum?id=aKkAwZB6JV

[28] Ali Yürekli, Alper Bilge, and Cihan Kaleli. 2021. Exploring playlist titles for cold-start music recommendation: an effectiveness analysis. *Journal of Ambient Intelligence and Humanized Computing* 12, 11 (2021), 10125–10144. doi:10.1007/s12652-020-02777-3

[29] Hongzhi Zhang and M. Omair Shafiq. 2024. Survey of transformers and towards ensemble learning using transformers for natural language processing. *Journal of Big Data* 11, 1 (2024), 25. doi:10.1186/s40537-023-00842-0