

Playlist Generation from Titles: Transformers vs. Large Language Models

ENZO CHAROLOIS-PASQUA*, ELÉA VELLARD*, YOUSRA REBBOUD, PASQUALE LISENA, and RAPHAEL TRONCY, EURECOM, France

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 of 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 ranking mechanism, for which we compare a voting system and an LLM-based ranker. 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**.

Additional Key Words and Phrases: Large Language Models, Playlist generation, NLP

ACM Reference Format:

Enzo Charolois-Pasqua, Eléa Vellard, Youssra Rebboud, Pasquale Lisena, and Raphael Troncy. 2026. Playlist Generation from Titles: Transformers vs. Large Language Models. *ACM Trans. Recomm. Syst.* X, X (October 2026), 29 pages.

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 playlist titles significantly influence users' expectations, playing a crucial role in their selection process [23]. Moreover, playlists, and even private ones, generally have

*Both authors contributed equally to this research.

Authors' Contact Information: Enzo Charolois-Pasqua, enzo.charolois-pasqua@eurecom.fr; Eléa Vellard, elea.vellard@eurecom.fr; Youssra Rebboud, youssra.rebboud@eurecom.fr; Pasquale Lisena, pasquale.lisena@eurecom.fr; Raphael Troncy, raphael.troncy@eurecom.fr, EURECOM, Sophia Antipolis, France.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2770-6699/2026/10-ART

<https://doi.org/>

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 [19]. For these reasons, the playlist title can be a key element in recommender systems, particularly when the title reveals a commonsense theme rather than being personal [44]. 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 titles such as those mentioned above may already be covered by ready-made playlists offered by streaming services, users may instead want playlists titled “Housewarming Party”, “Spring Awakening”, or “Country Summer”¹, which do not have a straightforward one-to-one match with existing categories.

Large Language Models are increasingly being incorporated into research, including the field of recommender systems, as shown in recent work, to capture user preferences [32, 42] or refine the recommendation after the candidate selection [24, 26]. A broader reflection on the impact of LLMs in music recommendation has been recently proposed [13].

In this paper, we specifically address this challenge by proposing a pipeline for playlist generation using only a title as input. Unlike previous works [14, 27], 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-tuned 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 [7, 15]. Track recommendations for a given playlist title are subsequently made using similarity metrics and a ranking mechanism. For the latter, both a voting system and an LLM-based ranker are compared. In addition, we assess the capacity of Large Language Models (LLM) in playlist generation from titles, both in a zero-shot scenario and for ranking a pre-selection of tracks.

This work extends the finding presented in [3], making the following contributions:

- 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;
- An extension of the title-based pipeline to playlist continuation using one seed track, evaluated through two ranking strategies (new contribution);
- An assessment of the capabilities of prompt-based LLM in the task of ranking the selected candidate tracks (new contribution);
- A demonstration platform to use the system and generate, listen and save playlists from a given title (new contribution).

The methodological contribution lies in the formulation and systematic evaluation of an end-to-end title-based recommendation framework, rather than in the introduction of a new embedding, clustering, or ranking algorithm. Accordingly, this study addresses the following research questions:

RQ1 To what extent can a playlist title alone provide a useful signal for playlist generation in a zero-seed scenario?

RQ2 How do retrieval-based generation, prompt-based LLM generation, and LLM-based ranking compare in terms of quantitative and qualitative performance?

¹These are real playlist titles that one can find in the Million Playlists Dataset.

RQ3 To what extent can the title-based pipeline be extended to playlist continuation by incorporating a seed track?

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, which are also illustrated in the demonstration application, described in Section 5. Finally, Section 6 concludes the paper, highlights its limitations and outlines potential future work. The code and data of all our experiments are available at <https://github.com/elea-vellard/LLM-Playlist-Recommender>, while the demonstrator is available at <https://playlist-recommendation.tools.eurecom.fr/>.

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 [10, 21]. In 2024, an online web application used a combination of GPT-4 [30] 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 [7]. 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 [15], 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 [25] 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 [17].

In 2018, the annual RecSys Challenge focused on playlist completion, leading to the release of the Million Playlist Dataset by Spotify [5]. This dataset remains one of the most significant resources for the task today. The inclusion of zero-seed playlists in the challenge pushed several participants to adopt *cold start* strategies 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 [14], 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 [27], 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 [22], another approach relying on an 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 performance in the zero-seed scenario.

Other textual information is used in the literature to generate or to continue playlists. In [16], 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.

²<https://www.playlistnameai.com/>

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

This contextual information is however manually annotated. Emotions are leveraged in [28], extracted with an 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 [8] 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 [31] and TALKPLAY [9] perform a supervised fine-tuning of LLMs on pairs including a query (or conversation) and a list of recommended tracks. Conversational Music Recommendation was also the topic of the RecSys Challenge 2026 [11].

Recent research has increasingly explored the integration of large language models (LLMs) into recommender systems from multiple perspectives. *K-RagRec* [40] uses structured knowledge-graph information to improve LLM-based recommendation by reducing hallucinations and injecting up-to-date domain knowledge. *Lasso* [6] uses an LLM as a user-behavior simulator to capture cross-domain preferences. LLMs are used for encoding both collaborative-filtering (CF) signals and semantic item descriptions in *LLM2Rec* [20] and *MoRE* [33]. *USB-Rec* [41] is a conversational recommendation framework that trains and fine-tunes LLMs via a user-simulator RL approach: it constructs a synthetic RL training set by generating preference-optimal dialogues, then applies a self-enhancement strategy at inference.

These studies do not directly engage with titles, which distinguishes them from our work. Indeed, following the RecSys Challenge 2018, limited attention has been devoted to playlist titles, leaving the state of the art largely unchanged while transformer-based approaches and LLMs now enable systems to go one step further. To the best of our knowledge, this is the first work since the RecSys Challenge 2018 to revisit playlist titles as a primary signal (rather than auxiliary metadata) in a recommendation task, while other works focused on analysis [44].

3 Method

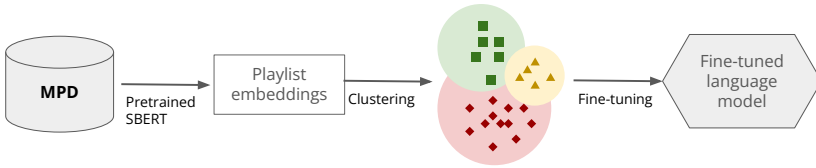


Fig. 1. Diagram representing our training approach

Our approach to playlist generation exploits the textual information of playlists (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 playlists and their 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 better able to capture semantic similarity in this specific context.

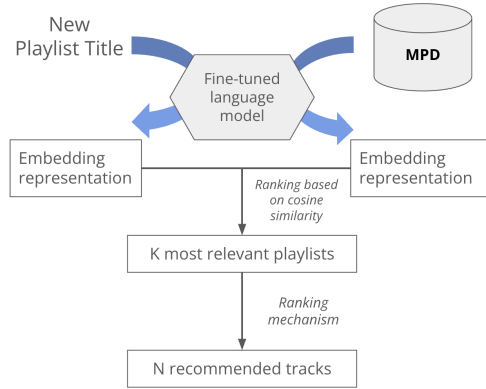


Fig. 2. Diagram representing our recommendation approach when a new playlist title is used as a seed

In the **generation 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 ranking 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 those that occur most frequently.

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

The proposed method relies on three main assumptions:

- playlist and track titles provide meaningful semantic proxies for the playlist theme;
- averaging the embeddings of track titles preserves the dominant theme of a playlist;
- playlists with semantically similar titles are likely to contain mutually relevant tracks.

The approach is consequently most suitable for thematically coherent playlists whose titles express an identifiable intent. Its effectiveness may decrease for personal, generic, or ambiguous titles, heterogeneous playlists, and cases in which relevance depends on information not expressed by the title, such as individual listening preferences, acoustic properties, or track ordering. These steps are further detailed in the following sections.

3.1 Processing the dataset

The Million Playlist Dataset (MPD) released by Spotify in 2018 [5] 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 artists, 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 easy-to-read 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 records 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

The design of the pipeline follows the distributional hypothesis, according to which items occurring in similar contexts tend to exhibit related meanings. In our setting, a playlist provides a domain-specific context: its track titles describe its musical content, while its title provides a concise linguistic description of the intended theme. Playlists containing semantically related tracks can therefore provide weak supervision for learning domain-specific representations of playlist titles.

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. The central hypothesis is therefore not that every playlist title is lexically similar to the titles of its tracks, but that repeated associations between playlist titles and playlist content provide a useful signal for learning domain-specific thematic similarity. We evaluate this hypothesis operationally by comparing the pre-trained and fine-tuned encoders on held-out playlist titles and on the downstream recommendation task.

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 [14], was deemed impractical due to its excessive size and computational inefficiency. The skip-gram approach, as described in [27], 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 preliminary qualitative assessment suggested that this model was less suitable for our use case, as it focuses primarily on word-level representations and does not explicitly model the contextual meaning of short titles. Figure 3 provides an illustrative visualization of this initial experiment.

Thus, we opted for a transformer-based model, which has demonstrated strong performance across a wide range of tasks [45]. 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 [34]. This enables Sentence-BERT to create more effective representations at

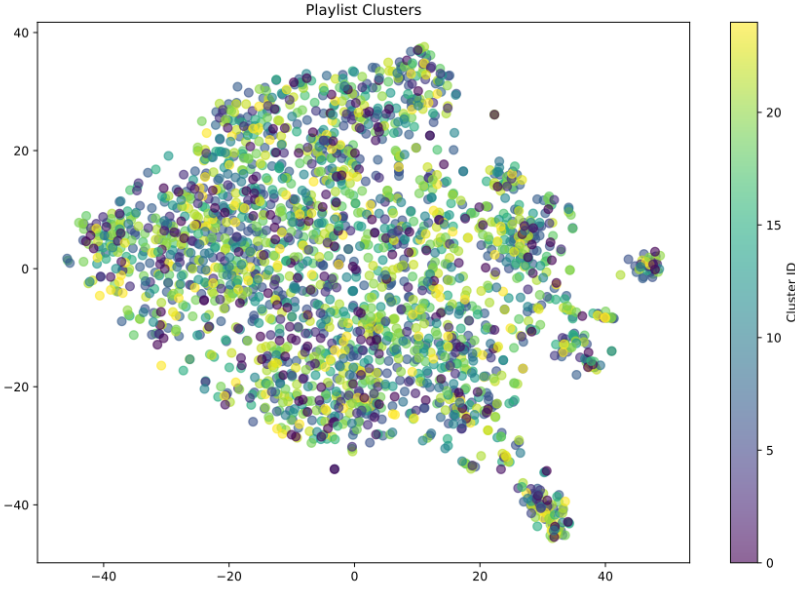


Fig. 3. T-SNE plot of 10 clusters for the training set using K-Means on fastText embeddings

the sentence-level. Given that playlist titles can be viewed as concise sentences (or at least are more extensive than single words), we therefore selected Sentence-BERT as a suitable model 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 that in fact represent 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 explore a suitable 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.

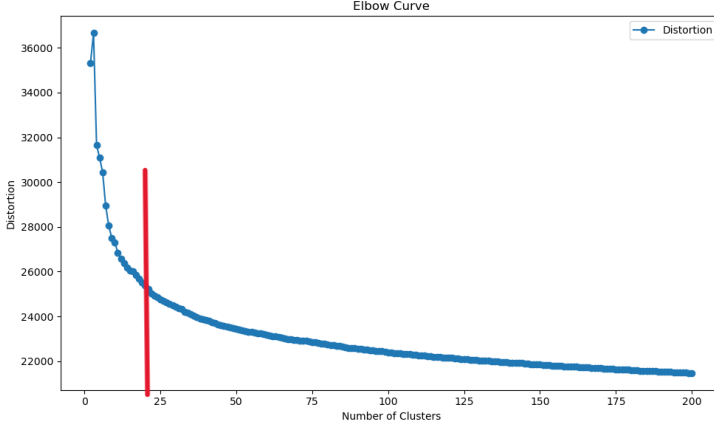


Fig. 4. The elbow curve and the suggested number of clusters (in red)

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 progressively reduced the within-cluster dispersion, as shown by the SSE values in Figure 5, and produced more thematically homogeneous clusters upon inspection. However, the marginal reduction becomes smaller as k increases, while a larger number of clusters also increases the complexity and computational cost of the subsequent fine-tuning task. We therefore selected $k = 200$ as a practical trade-off between obtaining finer-grained, more coherent clusters and keeping the training process computationally manageable. Therefore, based on our analysis, we settled on 200 clusters as a practical configuration for our experiments.

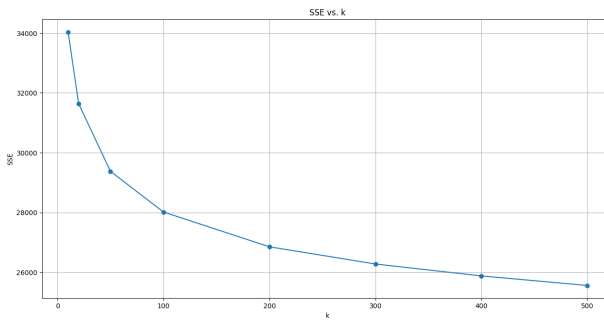


Fig. 5. The sum of squared errors (SSE) decreases as the number k of clusters increases.

The T-SNE plot of an initial experiment conducted 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 topics⁴, which we will refer to as *miscellaneous* throughout this paper.

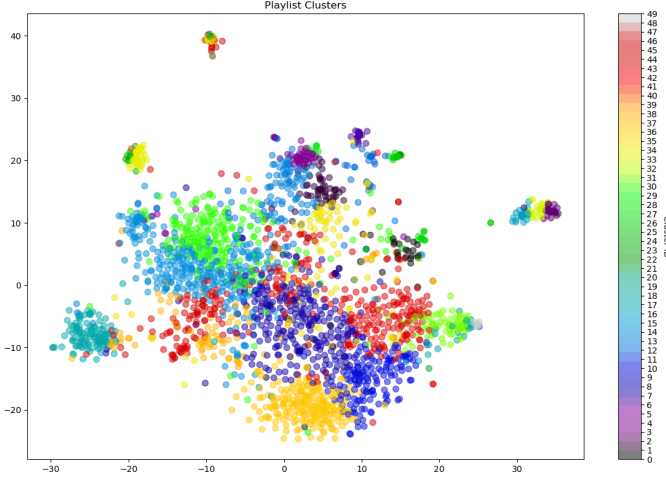


Fig. 6. T-SNE plot of 50 clusters for the training set using K-Means on Sentence-BERT embeddings. It is possible to note at least two *miscellaneous* clusters, in dark blue and green.

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, for each cluster we found the most frequently occurring playlist title and we computed the percentage of exact matches with it among all the playlists' titles of that cluster. We consider clusters with a percentage above the threshold t valid and those below it miscellaneous. The threshold has been set to $t = 0.02$ empirically. Most clusters below this percentage include a number of playlists well 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 remaining for the next sections.

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. The presence of miscellaneous playlists remains a limitation of our approach, as highly generic or non-informative titles provide insufficient semantic evidence for generating meaningful recommendations. However, this limitation is inherent to title-based recommendation systems more generally, since no method relying primarily on titles can reliably infer a playlist's intended theme when the title itself does not express it.

⁴In particular, the presence in the MPD of some broad-sense, not-meaningful playlist titles, such as "music" or "my playlist" was already reported in [14].

We split our processed dataset into training (80% of the data), validation (10%) and test (10%) sets, 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 sets created in the previous step are 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 act as documents in our fine-tuning, while cluster IDs are used as labels.

Concerning the architecture, we used a sentence transformer backbone [34] and in particular the all-MiniLM-L6-v2⁵ 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 [35].

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.

For the resulting model, we observed substantial 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 model exhibits less variability in its vectors, indicating a more uniform and compact embedding

⁵<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

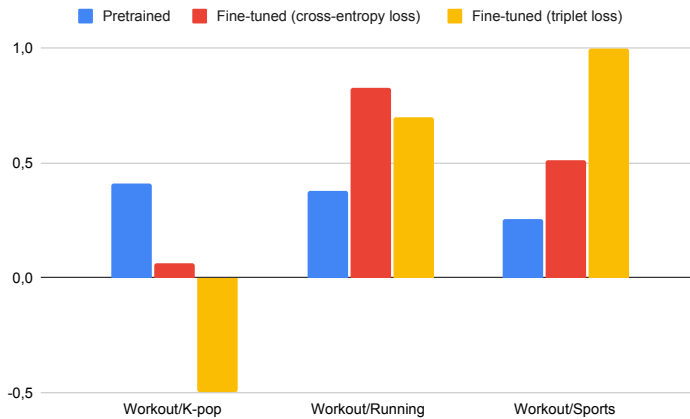


Fig. 7. The cosine similarity scores of pairs of embeddings computed with the three model variants.

space where embeddings 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 clearly within the semantic realm of physical activity, “Running” can also appear in metaphorical contexts – such as “running away from feelings” – and this may have 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 not exact matches with the proposed title.

The embeddings of each (known) playlist title in the dataset are computed using the fine-tuned model. As a seed for the generation process, an unknown playlist title – that can be composed of one or more 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:

$$\text{cosine_similarity}(e_a, e_b) = \frac{e_a \cdot e_b}{\|e_a\| \|e_b\|}$$

The $K = 50$ playlists with the highest similarity scores are selected, ready to be used as input to the **ranking mechanism**. Initially, we use a **voting mechanism** for ranking that works as follows. We count the occurrences of each track in the K selected playlists; tracks that appear more frequently across the matched playlists receive higher scores, indicating their relevance to the input theme. The top N most frequently occurring songs are then recommended to the user.

3.5 LLM-based ranker

As an alternative to the voting mechanism, which inherently favors popular songs, we propose a module utilizing a large language model to determine the final song ranking. This module shares the same input and output as the voting system. It takes the tracks from the top K most relevant playlists, then ranks these input tracks by relevance and outputs the N most relevant ones according to its parametric knowledge (recommended tracks).

The LLM is interrogated using LangChain [4], which also takes care of the parsing of the output. The prompt used is:

Prompt: LLM ranker

You are given a playlist title and a list of songs (song title and artist only).
Your task is to re-rank all songs based on their relevance to the playlist title.
Rank the songs from best fitting (rank 1) to least fitting.

Playlist title: {playlist title}

Songs: {list of songs with artists}

[format instructions]

Strictly follow the instructions. Never include null values. Take care of including the list of songs under the ‘songs’ property. Limit your list to 10. Output just the output JSON, without any other text. Do not add any note.

3.6 Playlist continuation with a seed track

We extend the proposed title-based generation pipeline to playlist continuation by providing one initial track, referred to as the *seed track*, together with the playlist title. Candidate playlists are retrieved exactly as in the original pipeline, using the cosine similarity between the input title and the titles of known playlists. The seed track is subsequently introduced during the candidate-song ranking phase. We investigate two strategies.

Seed-aware weighted voting. The first strategy modifies the original voting mechanism by assigning a higher contribution to playlists that contain the seed track. Let $\mathcal{P}_K(q)$ be the set of the K playlists retrieved for the input title q , and let z be the seed track. The score assigned to a candidate track s is:

$$\text{score}_{\text{vote}}(s \mid q, z) = \sum_{p \in \mathcal{P}_K(q)} w(p, z) \mathbf{1}_{\{s \in p\}}. \quad (1)$$

where:

$$w(p, z) = \begin{cases} 1, & \text{if } z \in p, \\ \beta, & \text{otherwise.} \end{cases} \quad (2)$$

Thus, tracks appearing in playlists that also contain the seed receive a full vote, while tracks appearing in playlists retrieved only through title similarity receive a lower contribution. The parameter β controls the contribution of playlists that are relevant to the input title but do not contain the seed track. We evaluate two configurations, $\beta = 0$ and $\beta = 0.3$. With $\beta = 0$, only playlists containing the seed contribute to the candidate-track scores. With $\beta = 0.3$, the remaining title-relevant playlists retain a lower contribution, preserving part of the broader thematic evidence provided by the playlist title.

Seed-similarity ranking. The second strategy uses the seed track directly to rank the candidate songs. After retrieving the K most relevant playlists from the input title, we collect their tracks and compute the cosine similarity between the embedding of each candidate track title and the embedding of the seed-track title. Candidate tracks are then ranked according to this similarity score:

$$\text{score}_{\text{sim}}(s \mid z) = \cos(e_s, e_z), \quad (3)$$

where e_s and e_z are the embeddings of the candidate-track title and the seed-track title, respectively.

For both strategies, the seed track itself is excluded from the final recommendations.

3.7 Playlist generation using LLMs

We also conducted a playlist generation experiment using solely a Large Language Model, leveraging again LangChain [4]. In particular, we tested one 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 (2017).

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 the model 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: Few-shot 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 denoted by S . The metrics we focus on are computed for a recommendation list of N songs [5, 37]:

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

$$precision@N = \frac{len(R \cap S)}{len(S)} \quad (4)$$

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

$$recall@N = \frac{len(R \cap S)}{len(R)} \quad (5)$$

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

$$MRR@N = \frac{1}{\text{Rank of the first relevant song in } S} \quad (6)$$

- **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 .

$$\text{R-precision} = \frac{|S_T \cap G_T| + 0.25 \cdot |S_A \cap G_A|}{|G_T|} \quad (7)$$

- **NDCG** (Normalized Discounted Cumulative Gain) evaluates the ranking quality of recommended tracks, improving as relevant tracks are positioned higher in the list. Over the N recommended songs, let $r_i \in \{0, 1\}$ indicate whether the track recommended at position i belongs to the ground-truth playlist, and let $IDCG$ be the DCG obtainable when the recommended playlist matches perfectly the ground truth:

$$DCG = \sum_{i=1}^N \frac{r_i}{\log_2(i+1)} \quad (8)$$

$$NDCG = \frac{DCG}{IDCG} \quad (9)$$

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. The latter serves as a general-purpose dense semantic retrieval baseline: it retrieves playlists through cosine similarity between title embeddings without any adaptation to the playlist domain. The comparison with the fine-tuned variants therefore isolates the contribution of domain-specific fine-tuning while keeping the remaining retrieval and ranking pipeline unchanged.

For each model, we computed metrics on the first 10, 66, and 500 tracks for each input playlist title, where 66 is the average number of tracks per playlist across the whole dataset, and 500 is the number of tracks requested in the RecSys Challenge 2018.

Table 1 shows the average metrics computed over multiple input playlists. Fine-tuning generally improves performance over the pre-trained model, although the magnitude of the improvement depends on the recommendation depth. The largest gains are observed at $N = 66$, where the cross-entropy variant improves Precision, Recall, and R-Precision by more than 100% compared with the pre-trained model. The two fine-tuned variants obtain broadly similar results, although the cross-entropy variant performs better across most metrics, while the triplet-loss variant generally achieves lower scores. As the number of recommended tracks increases, however, the differences between the variants become smaller. At larger depths, popular tracks are more frequently selected because they occur across many playlists, which progressively reduces the influence of differences in the retrieved playlist sets.

At $N = 500$, this convergence is particularly visible for the pre-trained and fine-tuned cross-entropy variants, which obtain almost identical R-Precision and MRR, with slightly lower values for the fine-tuned model and more pronounced for the triplet-loss variant. This can also be related to the separation between playlist retrieval and track ranking in our pipeline. Fine-tuning influences which playlists are retrieved through title similarity, whereas the subsequent voting mechanism discards their similarity order and ranks tracks only according to their frequency across the selected playlists. Moreover, the original track order within each playlist is neither used during training nor preserved during ranking. As a result, improvements in semantic playlist retrieval do not necessarily translate uniformly to every track-level metric, particularly at larger recommendation depths.

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
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)

In Table 2, we compare our strategy with those in previous works. The figures for the methods in [14, 22, 27] 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 related 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 the strongest title-only competitors by several points. In terms of NDCG, our method also outperforms some of the best variants that incorporated additional information about the tracks in the training.

4.2 Evaluation of playlist continuation

We evaluate the two playlist-continuation strategies described in Section 3.6 using one seed track in addition to the playlist title. For the seed-aware weighted-voting strategy, we evaluate two values of the parameter β : $\beta = 0$, where playlists that do not contain the seed do not contribute to the ranking, and $\beta = 0.3$, where they retain a reduced contribution. We also evaluate the seed-similarity strategy, in which candidate tracks are ranked according to the cosine similarity between their

Method	R-Precision	NDCG
Monti et al. [27]		
- <i>Only title</i>	0.0837	0.1260
- <i>Best</i>	0.1634	0.1717
Faggioli et al. [14]		
- <i>Only title</i>	0.1093	0.2451
- <i>Best</i>	0.2078	0.3713
Kim et al. [22]		
- <i>Only title</i>	0.0760	0.1866
- <i>Best</i>	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

titles and the seed-track title. Table 3 compares these configurations with the title-only fine-tuned model using cross-entropy loss.

Metric	Title only	Seed-aware weighted voting		Seed similarity
		$\beta = 0$	$\beta = 0.3$	
Precision@10	0.1793	0.1728	0.1728	0.0476
Recall@10	0.0382	0.0387	0.0387	0.0096
MRR@10	0.3254	0.3040	0.3040	0.1152
R-Precision@10	0.0496	0.0486	0.0486	0.0148
NDCG@10	0.3740	0.3422	0.3422	0.1467

Table 3. Comparison of title-only playlist generation and two playlist-continuation strategies using one seed track. The two tested values of β obtain identical aggregate results.

The seed-aware weighted-voting strategy performs substantially better than ranking candidate tracks solely according to their similarity with the seed track. Its performance also remains close to that of the title-only pipeline. Recall increases slightly, while other metrics decrease moderately. This suggests that prioritizing playlists containing the seed helps retrieve a small number of additional relevant tracks, although it does not improve the overall ranking.

The two tested values of β produce identical results because they yield the same top-10 recommendation lists in our evaluation. The parameter β only affects the contribution of retrieved playlists that do not contain the seed track. With $\beta = 0.3$, tracks from these playlists accumulate reduced weighted votes, but these contributions are not sufficient to move them above the top-10 cutoff. The highest-ranked tracks are already dominated by playlists containing the seed, whose contributions retain a weight of 1. Consequently, setting β to either 0 or 0.3 does not alter the final recommendations under the evaluated configuration.

The seed-similarity strategy obtains markedly lower scores across all metrics. Semantic similarity between song titles appears to be a weak proxy for musical compatibility: tracks that fit the same playlist may have semantically unrelated titles, while songs with similar titles may belong

to different genres, moods, or listening contexts. Moreover, this strategy does not exploit the aggregated track co-occurrence evidence captured by the voting mechanism.

Overall, the results demonstrate that the proposed architecture can be extended to playlist continuation without retraining the language model. Seed-aware weighted voting provides a more effective integration of title and seed information than direct similarity between song titles, although it does not outperform the original title-only configuration.

4.3 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 4, selected according to the following criteria:

- 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 sufficiently representative of the playlist content;
- both niche and more broadly themed playlists should be included in the final set.

This subset is intended to support the human assessment of thematic coherence and is not assumed to be statistically representative of the complete test 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 4. 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 independently assessed by two reviewers as fitting or not fitting the theme expressed by the title. In cases of disagreement, the reviewers

discussed the track and reached a consensus. The final qualitative score was computed from these consensus judgments. 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 would be considered not fitting. The assigned qualitative score is the percentage of valid tracks over the total, computed as:

$$\text{Qualitative Score} = \frac{\text{nb. valid tracks}}{\text{total recommended tracks}} \quad (10)$$

The metric is in the $[0, 1]$ range, being 1 when all recommended tracks are valid. Since the final labels were obtained through consensus, no inter-annotator agreement statistic is available for the reported judgments.

The results in Table 5 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 strong qualitative scores 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-entropy.

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 5. Qualitative 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).

4.4 Evaluation of the few-shot LLM generation

Model	Llama			Zephyr			GPT-4o			Gemma 4
n-shot	0	1	5	0	1	5	0	1	5	5
Precision@10	0.0591	0.0950	0.0636	0.0227	0.0409	0.0590	0.0636	0.1091	0.1227	0.1500
Recall@10	0.0067	0.0173	0.0059	0.0019	0.0060	0.0103	0.0073	0.0122	0.0197	0.0250
MRR@10	0.0966	0.1883	0.1723	0.0795	0.1318	0.1800	0.1636	0.2871	0.2505	0.2792
R-Precision@10	0.0137	0.0269	0.0104	0.0062	0.0103	0.0171	0.0157	0.0219	0.0338	0.0441
NDCG@10	0.1199	0.2102	0.1863	0.0952	0.1436	0.2011	0.1900	0.3039	0.3249	0.4095
Qualitative	0.6416	0.6818	0.7132	0.6777	0.6945	0.7038	0.7175	0.7724	0.7953	0.7909

Table 6. Performance metrics for LLM-based playlist generation. In **bold**, the best absolute scores; in **bold italics**, the best scores among zero-shot configurations.

We conducted an initial assessment of how LLMs perform using the same 22 playlists that were used for the qualitative evaluation. We tested four different LLMs, namely Llama 3.1 [12], Zephyr⁶ [39], GPT-4o [29], Gemma 4⁷ [38] in order to have representatives of both open-weight and proprietary models. While Llama 3.1, Zephyr, and GPT-4o were evaluated in zero-, one-, and five-shot settings, Gemma 4 (included at a later stage) was evaluated in the five-shot setting. The

⁶<https://huggingface.co/TheBloke/zephyr-7B-beta-AWQ>

⁷<https://huggingface.co/google/gemma-4-31B-it>, in reasoning mode

results are presented in Table 6. Although the quantitative metrics are based on a different test set from that used in Table 1⁸, this experiment can already give us some insight into LLM performance.

For the models evaluated under multiple prompting configurations, the inclusion of examples generally improves performance, although the effect is not uniform across models and metrics. Llama has a unique behavior, as it prefers 1 to 5 examples for most metrics, though not for qualitative assessments. Among the models evaluated in all three prompting configurations, GPT-4o obtains the strongest overall performance. However, the five-shot Gemma 4 configuration achieves the best scores in several metrics. In addition, we noticed that Llama and in particular Zephyr did not consistently follow the provided instructions, recommending songs published after the desired date specified in the prompt. GPT-4o 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 5), 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, GPT-4o and Gemma succeed in doing so in the 5-shot scenario.

Metric	22-playlist subset	Full test set
Precision@10	0.1500	0.1139
Recall@10	0.0250	0.0215
MRR@10	0.2792	0.2722
R-Precision@10	0.0441	0.0356
NDCG@10	0.4095	0.3280

Table 7. Performance of the five-shot Gemma 4 configuration on the selected 22-playlist subset and on the complete test set.

To assess whether the results obtained on the selected 22-playlist subset generalize to the broader dataset, we additionally evaluated the five-shot Gemma 4 configuration on the complete test set using the same generation and evaluation protocol. As reported in Table 7, performance is lower on the complete test set across all metrics. This was expected, since the 22-playlist subset was explicitly selected to contain titles with a clear thematic correspondence to their tracks. Nevertheless, the differences remain relatively limited for Recall@10 and MRR@10, and the results confirm that the model’s performance extends beyond the manually selected subset.

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.

4.5 Evaluation of the LLM ranker

The observed limitations of LLMs in few-shot generation may be primarily due to the inherent difficulty of identifying relevant candidates from a broad and unstructured space. For this reason, we investigate an alternative evaluation framework, providing LLMs with a pre-selected list of songs to rank, eliminating the challenge of candidate selection and isolating the ranking performance itself. In other words, we replace the voting mechanism with the LLM-based ranker described in Section 3.5. Our goal is to empirically determine which method performs more effectively.

⁸For reference, the pre-trained Sentence-BERT had Precision 0.2091, Recall 0.0440, and MRR 0.4307 on the same test set of 22 playlists.

We conducted an evaluation using a diverse set of state-of-the-art language models, chosen to represent a broad spectrum of architectural designs and performance capabilities. Our selection includes:

- open-weight models: **Gemma3** [18], **Llama3.1**, **Llama3.2** [12], **Qwen3** [43], **Zephyr** [39];
- closed-weight models: **GPT-4.1** [30], **GPT-4o** [29].

Metric	Simple						CoT		Voting
	gemma3	llama3.1	llama3.2	qwen3	gpt41	gpt4o	gpt41	gpt4o	Mechanism
Precision@10	0.0045	0.0050	0.0000	0.0000	0.1636	<u>0.1818</u>	0.1455	0.1773	0.2091
Recall@10	0.0002	0.0003	0.0000	0.0000	0.0310	<u>0.0418</u>	0.0217	0.0344	0.0440
MRR@10	0.0227	0.0050	0.0000	0.0000	0.2516	0.3422	0.2894	<u>0.3682</u>	0.4307
R-Prec@10	0.0029	0.0029	0.0039	0.0020	0.0423	<u>0.0542</u>	0.0318	0.0458	0.1453
NDCG@10	0.0287	0.0145	0.0000	0.0000	0.3114	<u>0.4092</u>	0.2989	0.4090	0.4682

Table 8. Performance of the LLM ranker for different models, comparing Simple and Chain-of-Thought (CoT) variants with the Voting Mechanism. In **bold**, the best score; underlined, the second best.

The results obtained on the subset of selected playlists (Table 4) indicate that the voting mechanism consistently outperforms LLM-based ranking across all evaluated metrics, as summarized in Table 8. While GPT-4o and GPT-4.1 follow closely behind, other models exhibit substantially lower performance. Notably, even advanced LLMs like GPT-4o occasionally introduce minor inconsistencies, such as the repetition of songs within the same playlist. Llama3.2 fails to generate valid outputs, instead repeatedly producing only the schema structure without meaningful content. Zephyr – which does not appear in the table given the poor results – and Qwen exhibit hallucinatory behavior, inventing songs not present in the candidate list. Qwen, in particular, further compounds this issue by repeating identical songs multiple times within a playlist – e.g., generating a playlist entirely composed of entries titled "You" by the artist "You". Such behavior underscores a failure to respect task constraints and ensure output diversity, even within a structured evaluation framework.

We additionally evaluated a Chain-of-Thought (CoT) prompting strategy by instructing the two best-performing models, gpt4.1 and gpt4o, to explicitly articulate their intermediate reasoning steps. To this end, we extended the original prompt with the instruction: “*Solve the following problem step by step and show your reasoning in the ‘reasoning’ property.*” However, the experimental results indicate that performance remained substantially unchanged: the obtained scores were comparable to those achieved with the simpler prompt formulation, with no statistically significant differences observed across evaluation metrics.

From a practical standpoint, the computational cost associated with deploying LLMs, relative to the voting mechanism, does not justify their use in this setting, given that the voting mechanism performs better across all evaluated metrics. However, it is important to acknowledge that variations in prompt design, particularly those emphasizing reasoning, could potentially yield different outcomes. Further exploration in this direction may provide additional insights into optimizing LLM-based ranking systems.

5 Demonstration Application

To better show the capabilities of our solution, we developed a web application for generating playlists using our method. This application is hosted at <https://playlist-recommendation.tools.eurecom.fr/>.

The **pipeline** of the demo is represented in Figure 8. The main components are:

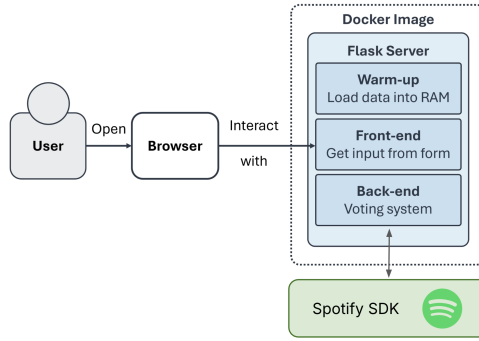


Fig. 8. Diagram representing the overview of the demo pipeline.

- a *back-end* that wraps the recommender system and communicates with the Spotify SDK⁹ for saving the playlists;
- a *front-end*, implemented with standard web technologies;
- a web server implemented in Flask that handles communication between the other components through API calls returning JSON results.

For portability, the system is provided as a self-contained Docker image based on `python:3.11-slim` with a footprint under 6 GB, including the machine learning model.

To guarantee fast response times and efficient retrieval during user interaction, the application performs a warm-up phase at startup. This procedure ensures that all critical components are preloaded into memory, minimizing computational overhead during runtime. Specifically, the system first loads the Sentence-BERT tokenizer, enabling immediate text encoding of any playlist title provided by the user. It then loads the fine-tuned model and initializes an index containing the precomputed embeddings of all known playlists in the dataset. This in-memory index allows for rapid nearest-neighbor retrieval using cosine similarity, making real-time recommendations possible with minimal latency¹⁰.

The **user interface** of the demo (Figure 9) is intentionally minimal and intuitive, designed to focus the user’s attention on the core functionality: playlist generation based on a title. The landing page presents a simple, clean layout with a central input field where users can freely enter a playlist title, expressed in natural language. The interface supports a wide range of inputs, from concise labels such as *workout* or *chill* to more complex, descriptive titles like *relaxing indie songs for studying on a rainy day*.

Upon submitting the title, the system displays a list of ten recommended tracks, each presented with the track name and artist. For each recommended song, a clickable Spotify icon is provided, enabling users to immediately listen to the track via Spotify’s web or desktop application. This direct playback option enhances the interactivity of the demo and facilitates real-time validation of the recommendations. Moreover, users can save the playlist to their Spotify profile and retain it in their collection. This feature also enables future real-user evaluation based on interactions collected through the platform.

The interface is lightweight, responsive, and refreshes instantly when a new title is provided. It is deliberately free of additional elements, sidebars, or complex controls, offering a focused

⁹<https://developer.spotify.com/>

¹⁰An average of 270 ms (min 177, max 392) over 10 distinct calls, including network latency.

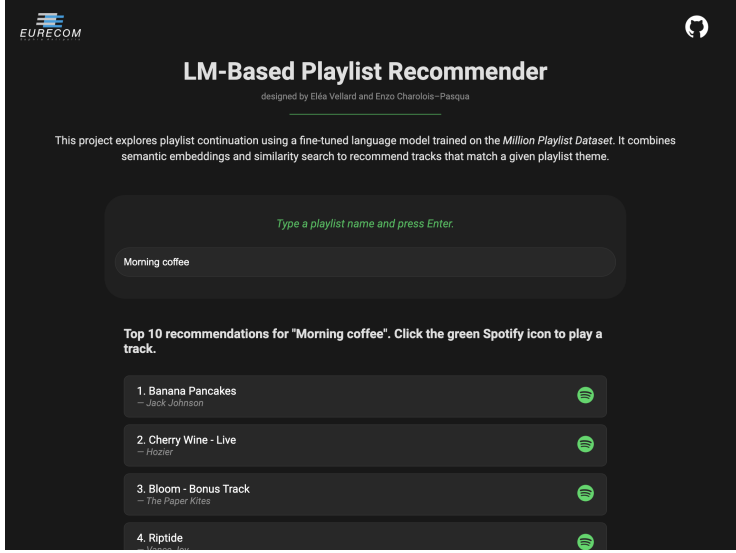


Fig. 9. The UI of the demo application

and seamless user experience. This design supports rapid experimentation and makes the system accessible to both technical and non-technical audiences.

6 Conclusion and Future Work

In this work, we study how language models can be used 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 approach outperforms previously reported title-only methods and the evaluated zero- and few-shot LLM configurations.

We further showed that the same architecture can be extended to playlist continuation by introducing a seed track during the ranking stage. Seed-aware weighted voting maintained performance close to the title-only configuration, whereas ranking candidate songs solely according to their semantic similarity with the seed produced substantially lower results.

Our approach empowers users to transcend the limitations of predefined categories, exploiting the full potential of their choice of words as a seed for generation. This opens up virtually unlimited combinations, offering users a novel way to discover relevant and potentially new and unexpected music.

6.1 Limitations

This work presents several limitations that necessitate future research. The playlist representation is based on the mean of the track-title embeddings and therefore captures primarily the dominant textual theme of a playlist. This compact representation discards complementary information such as artist identity, genre, and acoustic characteristics, as well as the ordering of tracks. While this choice is consistent with our focus on thematic recommendation from titles, it limits the ability of

the model to distinguish playlists with similar textual content but different musical profiles, and to represent transitions or sequencing patterns within a playlist.

Our implementation is designed to maximize retrieval-based metrics, as this was the evaluation focus of the RecSys Challenge 2018 and allows us to make meaningful comparisons with the state of the art. Introducing metrics that capture diversity, novelty, and order would undoubtedly enhance the evaluation. For example, to recommend novel songs not yet included in any playlist, it would be possible to incorporate Sentence-BERT similarity between a novel song title and existing playlist titles. This approach may result in lower precision metrics, as it prioritizes novelty over exact matches. The appropriate trade-off among the metrics would depend on the use case. In different scenarios, the relative superiority of the voting system may also change, potentially providing a more favorable setting for LLMs. Indeed, large language models demonstrated high qualitative scores, although we should acknowledge potential biases stemming from the evaluators' preferences [36].

The evaluation does not include a broader set of general text-retrieval baselines, such as sparse lexical retrieval or more recent text-to-item retrieval architectures. Although the pre-trained Sentence-BERT variant provides a general dense-retrieval baseline, a wider comparison would further contextualize the contribution. More generally, the proposed system models title-conditioned thematic relevance rather than personalized preference, since it does not consider user histories, acoustic features, or the intended sequence of tracks.

The voting mechanism tends to flatten differences for very common playlist titles and to favor popular tracks, particularly for generic titles. We investigated LLMs both as direct playlist generators and as rankers of pre-selected candidate tracks. Prompt-based generation achieved strong qualitative relevance, with Gemma 4 obtaining results close to the best evaluated configuration, although it remained below the fine-tuned retrieval pipeline on the complete test set. Similarly, the evaluated LLM-based rankers did not outperform the voting mechanism and introduced substantially higher computational costs.

We also evaluated two strategies for extending the system to playlist continuation using one seed track. Seed-aware weighted voting achieved results close to those of the title-only pipeline, with a slight improvement in recall and moderate reductions in the remaining metrics. In contrast, ranking candidate tracks solely according to the cosine similarity between their titles and the seed-track title performed substantially worse. This indicates that textual similarity between song titles is not a reliable proxy for musical compatibility.

6.2 Future Work

Future work should investigate richer mechanisms for combining title and seed information, including multiple seed tracks, learned weighting functions, and representations incorporating artists, acoustic characteristics, or track co-occurrence. Sequence-aware representations could also preserve track order and model transitions between songs. Ablation studies over these different information sources would help quantify their respective contributions to title-based generation and playlist continuation.

Alternative language models can also be explored as substitutes for Sentence-BERT to provide a more comprehensive comparison of their performance and capabilities. Specializing the model for specific languages could further 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 that the model can handle large volumes of data and user interactions seamlessly. In addition, we plan to include explainability mechanisms by providing users with insights into why

certain tracks were recommended. Finally, we believe that the value of this work should be tested in combination with collaborative recommendation methods, exploiting the strengths of both 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 as 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, Software, 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] Enzo Charolois-Pasqua, Eléa Vellard, Youssra Rebboud, Pasquale Lisena, and Raphael Troncy. 2025. A Language Model-Based Playlist Generation Recommender System. In *Proceedings of the 19th ACM Conference on Recommender Systems (RecSys '25)*. ACM, Prague, Czech Republic, 11. doi:10.1145/3705328.3748053
- [4] Harrison Chase. 2022. *LangChain*. LangChain AI. <https://github.com/langchain-ai/langchain>
- [5] 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
- [6] Yue Chen, Susen Yang, Tong Zhang, Chao Wang, Mingyue Cheng, Chenyi Lei, and Han Li. 2025. Lasso: Large Language Model-based User Simulator for Cross-Domain Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25)*. Association for Computing Machinery, New York, NY, USA, 207–216. doi:10.1145/3705328.3748048
- [7] 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.
- [8] 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>
- [9] Seunghoon Doh, Keunwoo Choi, and Juhan Nam. 2025. TALKPLAY: Multimodal Music Recommendation with Large Language Models. arXiv:2502.13713 [cs.LG] <https://arxiv.org/abs/2502.13713>
- [10] Seunghoon 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 Quadrona, Mohamed Sordo, and Kento Watanabe (Eds.). Association for Computational Linguistics, Online, 27–31. <https://aclanthology.org/2021.nlp4mus-a-1.6/>
- [11] Seunghoon Doh, Sergio Oramas, Bruno Sguerra, Abhinav Bohra, Claudio Pomo, and Francesco Barile. 2026. Overview and Analysis of the RecSys Challenge 2026: Conversational Music Recommendation. In *Proceedings of the Workshop on the ACM RecSys Challenge (RecSysChallenge '26)*. Association for Computing Machinery, New York, NY, USA, 1–9. doi:10.1145/3842413.3842414

- [12] 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, Jelier 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, et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024). doi:10.48550/arXiv.2407.21783
- [13] Elena Epure, Yashar Deldjoo, Bruno Sguerra, Markus Schedl, and Manuel Moussallam. 2026. Music Recommendation with Large Language Models: Challenges, Opportunities, and Evaluation. *ACM Trans. Recomm. Syst.* (Sept. 2026). doi:10.1145/3844509
- [14] Guglielmo Faggioli, Mirko Polato, and Fabio Aioli. 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
- [15] 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.
- [16] 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.
- [17] 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
- [18] Gemma Team et al. 2025. Gemma 3 Technical Report. arXiv:2503.19786 [cs.CL] <https://arxiv.org/abs/2503.19786>
- [19] Yun Hao. 2021. *Towards a better understanding of music playlist titles and descriptions*. Ph.D. Dissertation. University of Illinois at Urbana-Champaign.
- [20] Yingzhi He, Xiaohao Liu, An Zhang, Yunshan Ma, and Tat-Seng Chua. 2025. LLM2Rec: Large Language Models Are Powerful Embedding Models for Sequential Recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2* (Toronto ON, Canada) (*KDD '25*). Association for Computing Machinery, New York, NY, USA, 896–907. doi:10.1145/3711896.3737029
- [21] 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*. OpenReview, Washington DC, USA. <https://openreview.net/forum?id=nmtmjfJQLS>
- [22] 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
- [23] 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
- [24] Yueqing Liang, Liangwei Yang, Chen Wang, Xiong Xiao Xu, Philip S. Yu, and Kai Shu. 2025. Taxonomy-Guided Zero-Shot Recommendations with LLMs. In *Proceedings of the 31st International Conference on Computational Linguistics*, Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 1520–1530. <https://aclanthology.org/2025.coling-main.102/>
- [25] 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>
- [26] Changping Meng, Hongyi Ling, Jianling Wang, Yifan Liu, Shuzhou Zhang, Dapeng Hong, Mingyan Gao, Onkar Dalal, Ed Chi, Lichan Hong, Haokai Lu, and Ningren Han. 2025. Balancing Fine-tuning and RAG: A Hybrid Strategy for

- Dynamic LLM Recommendation Updates. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25)*. Association for Computing Machinery, New York, NY, USA, 919–922. doi:10.1145/3705328.3748105
- [27] 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
- [28] 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
- [29] 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 Codisoti, 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 Gierltler, 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 Aljube, 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 Bakum, 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 Cunningham, 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, Weiyei 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>
- [30] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, 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 Kopic, 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, Lillian 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>
- [31] 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).
- [32] Gustavo Penha, Edoardo D’Amico, Marco De Nadai, Enrico Palumbo, Alexandre Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, and Hugues Bouchard. 2025. Semantic IDs for Joint Generative Search and Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys ’25)*. Association for Computing Machinery, New York, NY, USA, 1296–1301. doi:10.1145/3705328.3759300
- [33] Weicong Qin, Yi Xu, Weijie Yu, Chenglei Shen, Xiao Zhang, Ming He, Jianping Fan, and Jun Xu. 2025. MoRE: A Mixture of Reflectors Framework for Large Language Model-Based Sequential Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys ’25)*. Association for Computing Machinery, New York, NY, USA, 299–308. doi:10.1145/3705328.3748055

- [34] 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
- [35] 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
- [36] Bruno Sguerra, Elena V. Epure, Harin Lee, and Manuel Moussallam. 2025. Biases in LLM-Generated Musical Taste Profiles for Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25)*. Association for Computing Machinery, New York, NY, USA, 527–532. doi:10.1145/3705328.3748030
- [37] Guy Shani and Asela Gunawardana. 2011. *Evaluating Recommendation Systems*. Springer US, Boston, MA, 257–297. doi:10.1007/978-0-387-85820-3_8
- [38] Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kasraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivièrè, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Brazinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Glenn Cameron, Charlotte Caucheteux, Garima Chadha, Jetha Chan, Aditya Chawla, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhuan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Takumi Fujimoto, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuzsak, Daniel McDuff, Gaurav Menghani, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Kate Olszewska, Bennett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikolaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, Benoit Seguin, Andrew Sellergrén, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Chetan Tekur, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiwei Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. 2026. *Gemma 4 Technical Report*. Technical Report. arXiv:2607.02770 [cs.CL] <https://arxiv.org/abs/2607.02770>
- [39] Lewis Tunstall, Edward Emanuel Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourier, 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*.

- <https://openreview.net/forum?id=aKkAwZB6JV>
- [40] Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge Graph Retrieval-Augmented Generation for LLM-based Recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 27152–27168. doi:10.18653/v1/2025.acl-long.1317
 - [41] Jianyu Wen, Jingyun Wang, Cilin Yan, Jiayin Cai, Xiaolong Jiang, and Ying Zhang. 2025. USB-Rec: An Effective Framework for Improving Conversational Recommendation Capability of Large Language Model. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25)*. Association for Computing Machinery, New York, NY, USA, 472–481. doi:10.1145/3705328.3748089
 - [42] Stanisław Woźniak, Jacek Duszko, Jan Kocoń, and Przemysław Kazienko. 2025. Improving LLM-Based Recommender Systems with User-Controllable Profiles. In *Companion Proceedings of the ACM on Web Conference 2025 (Sydney NSW, Australia) (WWW '25)*. Association for Computing Machinery, New York, NY, USA, 2102–2111. doi:10.1145/3701716.3717734
 - [43] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] <https://arxiv.org/abs/2505.09388>
 - [44] 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
 - [45] 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