Cold-Start Playlist Generation on the Spotify Million Playlist Dataset

Ozel Yilmazel, Samuel Lerner, Javin Mendiratta, Rishi Nandurbarkar, John Ryu

1 Abstract

Cold-start playlist generation is the task of automatically populating a playlist given only its title. This task presents a fundamental challenge in music information retrieval in the absence of user listening histories and due to the inherent vocabulary gap between playlist titles and track metadata. In this work we investigate multiple novel approaches, including transfer-based playlist initialization, multi-stage re-ranking, and a hybrid single-stage retriever that combines dense and lexical models. Our findings show the effectiveness of these new approaches and explores possible efficiency trade-offs. Additionally, we implement a new evaluation strategy to measure the musical feature overlap of playlists and retrieved tracks that captures semantic relevance instead of exact match of tracks.

2 Related Work

Automatic playlist continuation gained prominence through the Spotify Million Playlist Challenge [6]. The winning **vl6** system [4] employed multi-stage retrieval with matrix factorization for dense track-title representations followed by learning-to-rank re-ranking with extracted features. **Hello World!** [5] introduced separate neural modeling of titles (via character-level CNN) and playlist contents (via autoencoder), combining predictions linearly. **Avito** [3] similarly used dual matrix factorization (playlist-track and playlist-title) with re-ranking, demonstrating the effectiveness of modeling multiple modalities independently. **Creamy Fireflies** [1] showed that carefully tuned classical collaborative and content-based recommenders with strategic post-processing can compete with complex neural approaches. **HAIR** [7] achieved competitive performance through neighbor-based collaborative filtering with refined similarity functions and re-ranking, emphasizing efficiency over neural complexity. Following the challenge survey [6], we adopt standard evaluation metrics including precision and NDCG at various cutoffs. These works collectively demonstrate that effective playlist continuation relies on hybrid retrieval, multi-stage re-ranking, and representation learning techniques, principles that inform our approach.

3 Problem Statement

This project addresses the task of **cold-start playlist generation**, where the goal is to automatically create a relevant playlist given only its title as input. Specifically, given a playlist title p, the objective is to generate a ranked list of candidate songs that best match the intended theme, genre, or mood, ideally producing the top 100 tracks.

Each song s in the candidate pool is represented by a set of textual attributes, such as its track name, artist name, and album name. Using these features, we define a retrieval model M that assigns a relevance score between a playlist title and a song:

$$\text{score}(s \mid p) = M(s, p)$$

The cold-start problem is especially challenging because we lack listening histories or user-specific signals, and there's often a vocabulary gap between how tracks are named and how playlist titles are phrased, making this a difficult benchmark task.

4 Proposed Approach

We explored three model architectures that build upon lexical baselines through enriched representations and multi-stage retrieval strategies. All approaches utilize an **extended text representation** for tracks, constructed by concatenating the titles of all playlists containing each track in the training set. Formally, for a track s appearing in playlists $\{P_1, P_2, \ldots, P_k\}$, we define:

$$\text{text}(s) = \text{name}(s) \oplus \text{artist}(s) \oplus \text{album}(s) \oplus \bigoplus_{i=1}^{k} \text{title}(P_i)$$

, where $\oplus$ denotes concatenation. To prevent data leakage, we restrict $\{P_1, \ldots, P_k\}$ to training playlists only. Additionally, we employ zero-shot GPT-5 to annotate each playlist title with genre and thematic features (e.g., "chill," "workout," "indie rock"). For each track s, we aggregate features from all training playlists containing it:

$$\text{features}(s) = \bigcup_{s \in P_i} \text{features}(P_i)$$

All implementations of our approaches were original and written by our team.

4.1 Multi-Stage Re-ranker

This approach employs a two-stage pipeline combining lexical retrieval with learned re-ranking. In the first stage, we retrieve candidate tracks using Dirichlet prior query likelihood.

The top-1,000 retrieved tracks are then passed to a supervised listwise LTR re-ranker trained on the following feature vector:

- **Weighted Matrix Factorization Score:** $\mathbf{u}_p^\top \mathbf{v}_s$, where $\mathbf{u}_p$ and $\mathbf{v}_s$ are learned embeddings of playlist and track respectively in a shared latent space. This score comes from the baseline approach in *5.1.3*.
- **SVD Score:** Dot product $\mathbf{u}_p^{\text{SVD}} \cdot \mathbf{v}_s^{\text{SVD}}$ from singular value decomposition. This score comes from the baseline approach in *5.1.3*.
- **Popularity Score:** $|\{P : s \in P\}|$, the number of playlists containing s. This score was pre-computed based on the training set.
- **BM25 Score:** $\text{score}_{\text{BM25}}(s \mid p)$ based on extended text representation of the track. This score comes from the baseline approach in *5.1.2*.
- **Dirichlet Score:** $\text{score}_{\text{QL}}(s \mid p)$ from the first stage.
- **Track Text Length:** $|\text{text}(s)|$.
- **Track Duration:** Length in milliseconds.
- **Term Frequency:** $\sum_{w \in p} \text{tf}(w, s)$.

- **Feature Overlap:** $\frac{|\text{features}(s) \cap \text{features}(p)|}{|\text{features}(p)|}$, where s is the track and p is the playlist.

4.2 Hybrid Retriever

This single-stage approach combines complementary retrieval signals through linear interpolation. We define four scoring components:

(1) Lexical Matching: Dirichlet query likelihood $\text{score}_{\text{QL}}(s \mid p)$.

(2) Dense Retrieval: Using all-MiniLM-L6-v2 to encode titles and track text into dense vectors with retrieval via FAISS.

(3) Feature-Based Matching: Using annotated playlist features, we compute TF-IDF similarity normalized by feature diversity:

$$\text{score}_{\text{feat}}(s \mid p) = log \frac{1}{|\text{features}(s)|} \sum_{f \in \text{features}(p)} \text{tfidf}(f, s).$$

(4) Co-occurrence Score: For the top-1000 tracks from lexical retrieval, we compute pairwise co-occurrence frequency. For track s, we aggregate co-occurrence counts with all other highly-ranked tracks.

The final hybrid score normalizes each component to $[0, 1]$ and combines them via weighted interpolation:

$$\text{score}_{\text{hybrid}}(s \mid p) = \sum_{i=1}^{4} \lambda_i \cdot \text{score}_i(s \mid p), \quad \sum_{i=1}^{4} \lambda_i = 1,$$

where score_i denotes the normalized score for component i and λ_i are interpolation weights.

4.3 Transfer-Based Initialization

This novel approach addresses the cold-start problem by first retrieving similar playlists, then leveraging their tracks to initialize candidate selection.

Stage 1: Playlist Retrieval. We retrieve the top-N playlists most similar to query title p using weighted matrix factorization on the vocabulary space, derived from both playlist titles and track names, explained further in *5.1.3*:

$$P_{\text{sim}} = \arg \max_{P \in \mathcal{P}_{\text{train}}} \mathbf{u}_{pq}^{\top} \mathbf{u}_P,$$

where $\mathbf{u}_{pq}$ and $\mathbf{u}_P$ are latent representations of the query playlist title and a training playlist title respectively.

Stage 2: Track Retrieval via Transfer. From each retrieved playlist $P_i \in P_{\text{sim}}$, we randomly sample m tracks to form a transfer set $\mathcal{T} = \{s_1, s_2, \ldots, s_{N \cdot m}\}$. We compute an aggregate track representation via unweighted averaging:

$$\bar{\mathbf{v}} = \frac{1}{|\mathcal{T}|} \sum_{s_i \in \mathcal{T}} \mathbf{v}_{s_i},$$

where $\mathbf{v}_{s_i}$ is the latent embedding of track s_i derived from the term-frequency matrix $R \in \mathbb{R}^{(|T|+|P|) \times |V|}$. The scoring function is $score(s, v) = \bar{\mathbf{v}}^{\top} \mathbf{u}_s$, where $\bar{\mathbf{v}}$ and $\mathbf{u}_s$ represents the averaged embedding and track in the shared latent space. The construction of the term-matrix is explained further in *5.1.3.*

5 Experiments

5.0.1 Dataset. We conduct our experiments using the *Million Playlist Dataset (MPD)* [6]. Due to computational constraints, we use a sample of the dataset rather than the full corpus. From the original collection, we randomly sample 100,000 playlists and partition them into training, validation, and test splits comprising 70,000, 10,000, and 20,000 playlists, respectively. As the MPD does not include a predefined test set and the original challenge set used in the RecSys competition is no longer available and unmaintained, we adopt the common practice among prior studies of constructing our own evaluation splits. Consequently, all reported results and analyses are based on this sampled subset.

Our sampled subset maintains the same average playlist length (≈ 66 tracks per playlist) as the full dataset, thereby preserving the statistical properties relevant for retrieval evaluation. Across all splits, the sample contains 682,944 unique tracks and a vocabulary size of 181,782, derived from both playlist titles and track names.

5.0.2 Evaluation. We evaluate generated playlists by treating the original playlist tracks as the ground-truth relevant set. For each test playlist p, we use its contents as the relevance judgments (qrels). We adopt the partial credit scheme introduced in [6], which assigns a graded relevance score:

$$\text{rel}(s, p) = \begin{cases} 1.0 & \text{if } s \in p \text{ (exact track match)} \\ 0.25 & \text{if artist}(s) \in \{\text{artist}(s') : s' \in p\} \text{ (artist match)} \\ 0 & \text{otherwise} \end{cases}$$

This scoring function captures partial relevance when a recommendation matches the intended artist but not the exact track. Using these graded relevance scores, we compute several standard retrieval metrics:

- **Precision@k (P@5, P@10, P@100, P@R)**: Measures the proportion of relevant tracks among the top-k retrieved results, where *P@R* denotes R-Precision.
- **Mean Reciprocal Rank (MRR)**: Quantifies how early the first relevant track appears in the ranked list, averaged across all playlists.
- **Normalized Discounted Cumulative Gain (NDCG@5, NDCG@10, NDCG@100)**: Evaluates ranking quality while emphasizing correct ordering among retrieved tracks.

Feature Overlap Score (FO@10). To complement exact-match metrics, we introduce a novel evaluation measure based on feature-level overlap. Using our GPT-5 annotated features, we measure how well each retrieved track captures the semantic characteristics of the query playlist. For a playlist p with labeled features (features(p)) and a retrieved track list $\mathcal{R} = \{s_1, \ldots, s_k\}$, we define the feature overlap for each track:

$$\text{FO}(s_i, p) = \frac{|\text{features}(s_i) \cap \text{features}(p)|}{|\text{features}(p)|},$$

and compute the average across all retrieved tracks:

$$\text{FO@k}(p) = \frac{1}{k} \sum_{i=1}^{k} \text{FO}(s_i, p).$$

This metric measures the average proportion of playlist features captured by each retrieved track. Unlike exact track matching, feature overlap captures partial relevance at a semantic level. A retrieved track may not appear in the original playlist but still exhibit similar genre, mood, or thematic characteristics. This provides a more holistic evaluation particularly valuable for cold-start scenarios where diverse and thematically appropriate recommendations

Table 1: Performance comparison of all baseline models on the test set.

	P@5	P@10	P@100	P@R	MRR	NDCG@5	NDCG@10	NDCG@100	FO@10
Random	0.0010	0.0011	0.0011	0.0004	0.0008	0.0008	0.0008	0.0012	0.1790
BM25	0.0289	0.0296	0.0296	0.0267	0.0330	0.0260	0.0264	0.0398	0.5612
QL	0.0879	0.0820	0.0549	0.0617	0.1290	0.0852	0.0806	0.0844	0.7696
SVD	0.0073	0.0073	0.0099	0.0082	0.0070	0.0061	0.0062	0.0119	0.4953
WMF	0.0396	0.0392	0.0357	0.0343	0.0561	0.0361	0.0359	0.0516	0.8414
DPR	0.0239	0.0237	0.0181	0.0190	0.0246	0.0218	0.0216	0.0266	0.7619
Re-Ranker	**0.1068**	**0.0982**	**0.0622**	**0.0645**	**0.1630**	**0.1042**	**0.0976**	**0.0970**	0.8860
Hybrid	0.1018	0.0952	0.0609	0.0638	0.1449	0.0987	0.0938	0.0949	0.8348
Transfer-Based	0.0890	0.0834	0.0553	0.0548	0.1429	0.0851	0.0810	0.0825	**0.9384**

are desirable. A combined metric incorporating both exact match (e.g., NDCG) and feature overlap offers a comprehensive evaluation framework for future studies.

5.1 Baseline Models

5.1.1 Random Ranking. This baseline retrieves tracks uniformly at random from the corpus, serving as a performance lower bound.

5.1.2 Lexical Retrieval Baselines. We implement two classical text-based retrieval models using an inverted index over extended track representations (track metadata concatenated with training playlist titles):

Dirichlet Prior Language Model:

$$p(w \mid \theta_d) = \frac{|d| p_{\text{MLE}}(w \mid d)}{|d| + \mu} + \frac{\mu p_{\text{MLE}}(w \mid C)}{|d| + \mu}.$$

BM25:

$$\text{BM25}(q, d) = \sum_{t \in q \cap d} \log(IDF(t)) \cdot \frac{(k_1 + 1)\,\text{tf}(t, d)}{k_1 \left((1 - b) + b \frac{|d|}{\text{avgdl}}\right) + \text{tf}(t, d)}.$$

5.1.3 Matrix Factorization Baselines. Following [4], we construct a term-frequency matrix $R \in \mathbb{R}^{(|T|+|P|) \times |V|}$ where rows represent tracks and playlists, columns represent vocabulary terms, and entries indicate (i) how many playlists containing word w include track s, or (ii) term frequency of w in playlist title p. We define scoring for track as: $score(s, p) = \mathbf{u}_{pq}^{\top} \mathbf{u}_s$, where $\mathbf{u}_{pq}$ and $\mathbf{u}_s$ represents the query and track in the shared latent space. We apply two factorization methods:

SVD: Singular value decomposition via `scikit-learn`, projecting tracks and playlists into a shared latent space.

Weighted Matrix Factorization (WMF): Factorized by training to minimize errors in the reconstructed matrix.

5.2 Experimental Setup

For our experiments, we coded all the models, programs, and evaluation ourselves, relying on implementations from homeworks, and our previous milestones.

Pre-processing. For all text pre-processing we used NLTK word tokenizer, and lower cased tokens.

Baseline Hyperparameters. For Dirichlet smoothing, we set $\mu = 2000$ to combat vocabulary sparseness. For BM25, we use $k_1 = 1.2$ and $b = 0.75$. Both matrix factorization models (SVD and WMF) employ a latent dimensionality of $d = 200$. The WMF model trains for 50 iterations with L2 regularization $\lambda = 0.01$ and confidence parameter $\alpha = 30$.

Multi-Stage Re-ranker. We construct training data using Dirichlet baseline top-1,000 rankings as negatives, randomly sampling 40 negative examples per query. The XGBoost re-ranker is trained with listwise loss using the `rank:ndcg` objective.

Hybrid Retriever. We fine-tune the all-MiniLM-L6-v2 encoder using in-batch negative sampling following the DPR approach [2], with 20 hard negatives per query for 10 epochs. The final hybrid model combines four components with linear interpolation weights: $\lambda_{\text{lex}} = 0.4$ (Dirichlet with $\mu = 2000$), $\lambda_{\text{dense}} = 0.3$ (fine-tuned DPR), $\lambda_{\text{cooc}} = 0.15$ (co-occurrence), and $\lambda_{\text{feat}} = 0.15$ (feature overlap). The retrieval performance of the fine-tuned DPR is reported as DPR in 1 as another baseline.

Transfer-Based. We used top-20 playlists retrieved with WMF as the seed playlists, and randomly sampled 2 songs from each playlist to create the average track embedding.

6 Results

6.1 Results and Analysis

Table 1 presents the performance of all baseline and proposed models on the test set. We observe several notable trends across both traditional baselines and our proposed architectures.

6.1.1 Baseline Performance. Among the baseline models, the Dirichlet prior language model (QL) achieves the strongest performance across most metrics, with P@5 of 0.0879, MRR of 0.1290, and NDCG@10 of 0.0806, and WMF achieving the strongest FO@10 of 0.8414. The performance of the Dirichlet baseline is somewhat surprising given the simplicity of the approach, but can be attributed to the smoothing capabilities of the Dirichlet prior, which proves particularly effective in handling the sparse vocabulary inherent to our dataset. The vocabulary-based representations suffer from extreme sparsity with 181,782 unique terms distributed across 682,944 tracks, making proper smoothing essential for reliable retrieval.

BM25 performs substantially worse than QL, suggesting that its term saturation mechanism is less suited to this sparse setting than Dirichlet smoothing. Among the collaborative filtering baselines, WMF outperforms SVD, likely due to its ability to perform the factorization more effectively with the weights. The fine-tuned DPR baseline yields surprisingly modest results, performing below even BM25. We attribute this to the limited contextual information available in short track metadata and playlist titles, which limits the model's ability to learn meaningful semantic embeddings in this domain.

6.1.2 Proposed Model Performance. Our proposed models achieve substantial improvements over the baselines. The Re-Ranker model attains the highest performance across most exact-match metrics, with P@5 of 0.1068, MRR of 0.1630, and NDCG@10 of 0.0976, representing a 21.5% relative improvement in P@5 and 26.4% improvement in MRR over the best baseline (QL). This demonstrates that the multi-stage pipeline effectively combines complementary signals: the initial QL retrieval provides strong lexical matching, while the learned re-ranker successfully integrates collaborative filtering scores (WMF, SVD), popularity signals, and feature overlap to refine the ranking.

The Hybrid retriever achieves remarkably competitive performance (P@5 of 0.1018, MRR of 0.1449, NDCG@10 of 0.0938), closely trailing the Re-Ranker while offering significant practical advantages. As a single-stage model with little supervised training required for fine-tuning DPR, Hybrid eliminates the computational overhead and training complexity of the two-stage pipeline. Simple linear interpolation of lexical, dense, feature-based, and co-occurrence scores performs nearly as well as a learned re-ranker, suggesting that straightforward signal combination is effective in this domain. Notably, the Hybrid model's P@5 of 0.1018 represents a 326% relative improvement over the zero-shot DPR baseline (0.0239), highlighting the importance of combining dense retrieval with complementary signals rather than relying on dense representations alone.

The Transfer-Based Initialization approach achieves competitive exact-match performance, performing comparably to the QL baseline while using a fundamentally different retrieval strategy. Most notably, this approach achieves the highest feature overlap score of 0.9383, substantially outperforming all other models including the Re-Ranker (0.8860). This indicates that retrieving tracks from similar playlists captures rich semantic information about genre, mood, and thematic coherence. While the Transfer-Based model does not achieve the highest exact-match scores, its superior feature alignment suggests that thematically similar tracks occur in thematically similar playlists. This represents a promising direction for future exploration, particularly in cold-start scenarios where semantic diversity may be more valuable than exact matching.

7 Member Contribution

All members contributed to forming the research question and the accompanying report. Main approaches and baselines were decided on collectively as a team.

7.0.1 Ozel Yilmazel. Implemented helper functions to process the dataset, build the inverted index, and the evaluation code. Developed baseline models, BM25, QL, SVD, WMF. Created the annotations for playlist features using GPT-5. Led the development of the Re-Ranker model and Hybrid model, also fine-tuning the DPR.

7.0.2 John Ryu, Samuel Lerner, Rishi Nandurbarkar, Javin Mendiratta. Developed baseline models for WMF. Condensed data file for training dataset. Co-developed the Transfer-Based Retrieval Model.

8 Miscellaneous

Repository is available here: https://github.com/oz03-hub/spotify-final

References

[1] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio Ferrari Dacrema. 2020. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. *CoRR* abs/2010.06233 (2020). arXiv:2010.06233 https://arxiv.org/abs/2010.06233

[2] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906 [cs.CL] https://arxiv.org/abs/2004.04906

[3] Vasiliy Rubtsov, Mikhail Kamenshchikov, Ilya Valyaev, Vasiliy Leksin, and Dmitry I. Ignatov. 2018. A hybrid two-stage recommender system for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 16, 4 pages. doi:10.1145/3267471.3267488

[4] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-stage Model for Automatic Playlist Continuation at Scale. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 9, 6 pages. doi:10.1145/3267471.3267480

[5] Hojin Yang, Yoonki Jeong, Minjin Choi, and Jongwuk Lee. 2018. MMCF: Multimodal Collaborative Filtering for Automatic Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 11, 6 pages. doi:10.1145/3267471.3267482

[6] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2018. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *CoRR* abs/1810.01520 (2018). arXiv:1810.01520 http://arxiv.org/abs/1810.01520

[7] Lin Zhu, Bowen He, Mengxin Ji, Cheng Ju, and Yihong Chen. 2018. Automatic Music Playlist Continuation via Neighbor-based Collaborative Filtering and Discriminative Reweighting/Reranking. In *Proceedings of the ACM Recommender Systems Challenge 2018* (Vancouver, BC, Canada) *(RecSys Challenge '18)*. Association for Computing Machinery, New York, NY, USA, Article 10, 6 pages. doi:10.1145/3267471.3267481