Fusing Semantic and Acoustic Features for Controllable Playlist Generation from Natural Language Prompts

Candidate Numbers
36587, 45686, 46524, 45016

Abstract

We present a content-based music recommendation system that generates personalized playlists from user-defined natural language prompts. Unlike collaborative filtering approaches that depend on historical user-item interactions, our model integrates heterogeneous song metadata including lyrics, genre, language, and a rich set of audio features into a unified retrieval framework. User prompts and song text descriptions are embedded using a pretrained SentenceTransformer. To enhance interpretability and better align with the capabilities of our SentenceTransformer, which is optimized for extracting meaning from full natural language sentences, we supplement raw lyrics with natural language sentences generated by combining bucketed acoustic feature descriptors (e.g., "high energy," "low valence") and categorical metadata (e.g., "language is English," "genre is rock"). Overall, categorical variables such as genre and language are encoded as binary vectors, and continuous acoustic features (e.g., tempo, energy, loudness) are normalized and projected via a multi-layer perceptron (MLP). All feature vectors are fused through a projection module into a shared latent space, where cosine similarity is used to rank songs against the prompt embedding. We developed models through an iterative, version-by-version approach; at each stage, we incorporated techniques such as fine-tuning the encoder, leveraging curated prompt–playlist pairs, and applying explicit filtering to enhance retrieval precision and user relevance. Our final system includes tunable parameters that allow users to dynamically adjust the importance of semantic, categorical, and numerical components. This work offers a flexible alternative to black-box recommender systems by providing greater transparency and control over recommendation logic.

1 Introduction & Background

Music recommendation systems used by platforms like Spotify and Apple Music typically rely on collaborative filtering, which infers user preferences from historical behavior. While powerful at scale, such systems often lack transparency, fail to capture nuanced user intent, and struggle with the cold-start problem when recommending lesser-known tracks. To address these issues, van den Oord et al. (2013) proposed deep content-based models that predict user-item factors directly from raw audio, while Oramas et al. (2017) demonstrated the effectiveness of combining text and audio data in a multimodal architecture to recommend new artists [1, 2]. These works highlight how integrating interpretable content can improve the flexibility and robustness of recommendation systems.

Recent advances have also enabled prompt-based recommendation, as seen in Palumbo et al. (2025), who introduced Text2Tracks, a system that maps free-form user queries to track identifiers using generative retrieval [3]. While such approaches excel at modeling intent, they offer limited transparency

and user control. In this report, we take a modular, interpretable approach to playlist generation. Using the SpotGen dataset [4], which includes 90,000+ songs with metadata, lyrics, and detailed audio features from Spotify and Genius, we allow users to generate playlists based on free-text prompts, while also customizing filters such as release year range, track and artist popularity, and feature relevance weights (semantic, genre, language, numeric). This creates a transparent, controllable alternative to traditional black-box recommenders, enabling flexible and meaningful playlist creation grounded in both content and user input.

2 Methodology & Experiments

2.1 Data Preprocessing

Before doing any modeling and constructing our embeddings, we prepared a unified song dataset, building on the SpotGen dataset, through the following steps:

1. **Table Merging & Cleaning:** Corrected misencoded lyrics, dropped irrelevant columns, and converted stringified lists (e.g., genre tags) into native Python lists. Merged various tables containing track, artist, lyrics, and album infomration—all coming from the SpotGen dataset—into one master table.
2. **Language Annotation:** Applied the `papluca/xlm-roberta-base-language-detection` [6] model to assign a language label to each track, with a secondary "heavy" pass on English lyrics using longer text snippets. Any residual misclassifications were manually corrected.
3. **Numeric Feature Scaling & Bucketing:** Selected fourteen audio metrics (e.g., acousticness, energy, tempo) and min–max scaled them to $[0, 1]$. Defined five equal-span buckets ("very low" through "very high") based on each feature's empirical distribution.
4. **Descriptive Sentence Construction:** Concatenated bucket labels with key text fields (song name, artist, playlist, release date, genre, lyrics) to create a new feature for each song called `sentence`, which holds a natural-language summary of the track based on its metadata and audio features (formatted this way to align with the input style best handled by our `all-MiniLM-L6-v2` SentenceTransformer).

2.2 Multimodal Embedding Architecture

We embed both user prompts and track information and characteristics into a shared 384-dimensional space by fusing a text branch and a numeric branch (see Figure 1):

- **Text Branch:** Each descriptive sentence about tracks stored in `sentence` is encoded by the pretrained `all-MiniLM-L6-v2` SentenceTransformer [5]:

$$E_{\text{text},i} = \text{SBERT}\big(\text{sentence}_i\big)\,.$$

- **Numeric Branch:** The fourteen-dimensional normalized feature vector $x_{\text{num},i}$ is projected into the same 384-dimensional space via a two-layer MLP followed by L2 normalization:

$$E_{\text{num},i} = \text{Normalize}\big(\text{MLP}(x_{\text{num},i})\big)\,.$$

- **Fusion & Retrieval:** We concatenate the two embeddings and apply a learnable fusion layer (linear $\to$ ReLU $\to$ LayerNorm $\to$ normalize):

$$E_{\text{song},i} = \text{Normalize}\big(\text{LayerNorm}(\text{ReLU}(W_{\text{fusion}}\,[E_{\text{text},i}; E_{\text{num},i}]))\big).$$

  A user prompt is encoded analogously using a neutral numeric mean vector in place of $E_{\text{num},i}$ to yield E_{prompt}. Finally, we rank songs by cosine similarity:

$$\text{sim}(p, i) = \frac{E_{\text{prompt}} \cdot E_{\text{song},i}}{\|E_{\text{prompt}}\|\;\|E_{\text{song},i}\|},$$

  and select the top-k tracks as the recommended playlist.

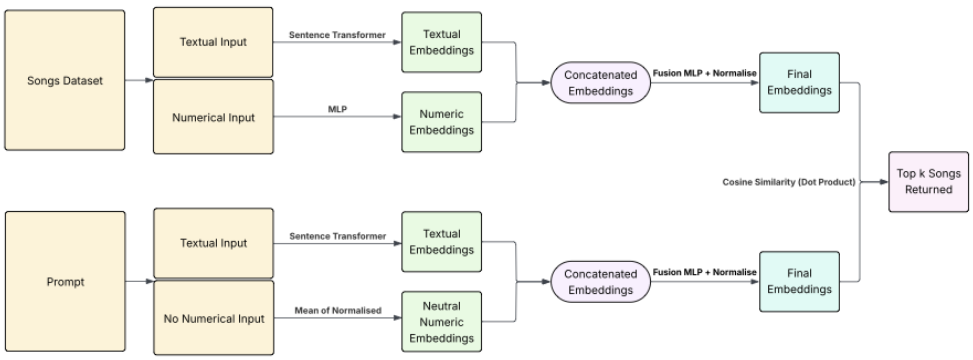

Figure 1: Workflow of Multimodal Embedding Architechture

2.3 Playlist Generation Retrieval Pipeline

As outlined earlier, we follow an iterative approach to building our prompt-to-playlist retrieval pipeline. We describe seven successive versions, each introducing new capabilities to better align user intent with song embeddings (see Table 1).

Base Version: Semantic Similarity

Our baseline system retrieves tracks by computing the semantic similarity between user prompts and tracks' representative sentences. Specifically, we embed both the prompt and each track's generated descriptive sentence using the `all-MiniLM-L6-v2` SentenceTransformer. Recommendations are produced by ranking tracks based on cosine similarity between these embeddings. This version does not use any additional features such as genre, popularity, or language.

Version 1: Explicit Genre Filtering

In Version 1, we honor explicit genre requests by first extracting all genre tags whose lowercase form appears in the user's prompt, pruning substrings to keep only the most specific ones. Let

$$\text{matched\_tags} \;=\; \big\{\, t \in \mathcal{T} \mid \text{lower}(t) \in \text{lower}(\textbf{prompt}) \big\}.$$

If $\text{matched\_tags} \neq \emptyset$, we restrict our candidate set to

$$\mathcal{S} \;=\; \big\{\, s \in \mathcal{D} \mid \exists\, t \in \text{matched\_tags} : t \in \text{genres}(s) \big\},$$

otherwise $\mathcal{S} = \mathcal{D}$. We then embed the prompt and each $s \in \mathcal{S}$ into our shared 384-dimensional space and compute

$$\text{score}(s) \;=\; \cos\big(E_{\text{prompt}},\, E_{\text{song},s}\big) \;\in\; [-1, 1].$$

Finally, we sort by $\text{score}(s)$ in descending order and return the top-k.

Version 2: Other User-Specified Filtering on Top of Genre Filters

Building on Version 1, we first apply user-specified filters for track popularity, artist popularity, and release year to narrow $\mathcal{D}$ to $\mathcal{S}_{\text{filtered}}$. Over this filtered set we compute the raw semantic similarity

$$\text{sim}(s) = \cos\big(E_{\text{prompt}}, E_{\text{song},s}\big) \;\in [-1, 1],$$

then linearly rescale to $[0, 1]$:

$$\widetilde{\text{sim}}(s) = \frac{\text{sim}(s) + 1}{2}.$$

We again extract and prune exact-match tags to form a binary genre flag $\text{genre\_flag}(s) \in \{0, 1\}$. The final composite score is

$$\text{score}(s) = \alpha\, \widetilde{\text{sim}}(s) \;+\; \beta\, \text{genre\_flag}(s), \qquad \alpha + \beta \leq 1.$$

We then select the top-k tracks by $\text{score}(s)$.

Version 3: Adding in Prompt-Song Pairs & Supervised Fine-Tuning

To improve the alignment between user prompts and recommended tracks, we adapt the pretrained `all-MiniLM-L6-v2` model to our playlist task via contrastive fine-tuning. This approach uses prompt–song pairs extracted from curated Spotify playlists to teach the model what types of songs best match particular queries.

We began by selecting six seed prompts—three music-themed (e.g., "Chill Hits") and three situational (e.g., "I'm cooking dinner")—and then expanded this list using ChatGPT-generated suggestions. Each prompt was mapped to a high-quality Spotify playlist containing at least 100 tracks and 1,000 followers. For each playlist, we retrieved track IDs via the Spotify Web API and paired the associated Version 2 sentence embedding with its corresponding prompt. This yielded a dataset of positive (prompt, track sentence) pairs. In-batch negatives were used to provide contrastive supervision during training.

Our fine-tuning procedure is based on the same `all-MiniLM-L6-v2` SentenceTransformer model used in Version 2. We optimize a cross-entropy loss applied to the cosine similarity matrix between prompts and songs, encouraging each prompt's true match to be ranked highest in its batch.

We trained the model using the Adam optimizer with a linear learning rate scheduler. Each batch contained 8 prompt–sentence pairs, and training was conducted for 10 epochs. To keep the search space manageable, we concentrated our hyper-parameter tuning on the learning-rate schedule: five candidates—1×10^{-4}, 5×10^{-5}, 1×10^{-5}, 5×10^{-6}, and 1×10^{-6}—were evaluated, and we chose the value that produced the lowest final average training loss.

All Version 2 song embeddings were then regenerated using this fine-tuned model to ensure better semantic alignment with user prompts during playlist retrieval.

Version 4: Language-Aware Matching

Building on Version 3, we parse the user's prompt for language keywords (e.g. "Spanish", "French"), handling negations (e.g. "not French") to yield a (possibly ordered) set of ISO codes $\{\ell_j\}$. For each song s, let

$$\text{lang\_flag}(s) = \begin{cases} 1, & \text{if } \text{lang}(s) \in \{\ell_j\}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\text{lang}(s)$ is the detected lyric language. We then extend the Version 3 composite score to

$$\text{score}(s) = \alpha\, \widetilde{\text{sim}}(s) + \beta\, \text{genre\_flag}(s) + \gamma\, \text{lang\_flag}(s),$$

with $\alpha + \beta + \gamma \leq 1$, and return the top-$k$ by $\text{score}(s)$. This ensures that, for example, "upbeat jazz in Spanish" prioritizes Spanish-language jazz tracks.

Version 5: Numeric Feature Boosting

Building on Version 4, we introduce a prompt-driven audio feature boost. Let $\mathcal{F}$ be the set of audio attributes referenced in the prompt (e.g. "dance" $\rightarrow$ danceability), and let each feature $x_{s,f} \in [0, 1]$ be its min–max scaled value for track s. Define

$$b_{\text{num}}(s) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} x_{s,f} \;\in\; [0, 1].$$

We then form the unified score

$$\text{score}(s) = \alpha\, \widetilde{\text{sim}}(s) + \beta\, \text{genre\_flag}(s) + \gamma\, \text{lang\_flag}(s) + \delta\, b_{\text{num}}(s),$$

subject to $\alpha + \beta + \gamma + \delta = 1$. Ranking by this score ensures that tracks matching both genre/language and the specified audio characteristics (e.g. "loud," "dance") are surfaced.

Version 6: Improving Fine-Tuning via Synthetic Playlists

After completing all previous model versions, we found that fine-tuning on real Spotify playlists—while beneficial—provided only modest improvements, since just 723 of the 7,441 requested tracks appeared in our metadata. To address this coverage gap and better capture complex

playlist semantics, we devised a two-stage regimen: first, we algorithmically generated roughly 98,000 synthetic playlists by intersecting songs across at least three categorical and numeric features (genre, decade, language, and top-20% audio traits such as danceability); then, we fine-tuned our SentenceTransformer on this vastly expanded, curated dataset. This pipeline not only overcomes the dataset's inherent limitations but also demonstrably enhances the model's ability to internalize playlist-level relationships, yielding significantly better alignment between user prompts and retrieved tracks without relying on rigid, hand-engineered filters.

Version	Description	Score Equation
Base	All songs; rank by SBERT cosine similarity.	$\cos\big(E_{\text{prompt}}, E_{\text{song},s}\big)$
v1	Hard-filter songs by any exact-match genre in prompt, then SBERT rank.	$\text{genre\_flag}(s) \times \cos\big(E_{\text{prompt}}, E_{\text{song},s}\big)$
v2	v1 + pre-filter by popularity, artist, year; soft genre bias.	$\alpha \, \dfrac{\cos(E_{\text{prompt}}, E_{\text{song},s}) + 1}{2} + \beta \, \text{genre\_flag}(s)$
v3	v2 + contrastive fine-tuning of the embedding model on prompt–song pairs from curated Spotify playlists to better capture user intent.	$\cos\big(E^*_{\text{prompt}}, E^*_{\text{song},s}\big)$ where E^* are fine-tuned embeddings
v4	v3 + binary language match.	$\alpha \, \tilde{\text{sim}}(s) + \beta \, \text{genre\_flag}(s) + \gamma \, \text{lang\_flag}(s)$
v5	v4 + audio feature boost (danceability, loud).	$\alpha \, \tilde{\text{sim}}(s) + \beta \, \text{genre\_flag}(s) + \gamma \, \text{lang\_flag}(s) + \delta \, b(s)$
v6	v5 + further fine-tuning on 98K synthetic playlists formed by intersecting genre, decade, language, and high-intensity audio traits.	$\alpha \, \cos\big(\hat{E}_{\text{prompt}}, \hat{E}_{\text{song},s}\big) + \beta \, \text{genre\_flag}(s) + \gamma \, \text{lang\_flag}(s) + \delta \, b(s)$, where $\hat{E}$ are embeddings fine-tuned on synthetic playlists

Table 1: Summary of Playlist-Generation Pipeline Versions

2.4 Learning Rate Selection via Cross–Validation

When training (using a fixed random seed of 42 across the notebook), we assembled approximately 5,000 prompt–sentence pairs (from only 620 unique prompt–song training examples) and fixed all other settings (Adam optimizer, batch size $B = 32$, no weight decay or dropout). Given the modest size of our dataset, we chose 5-fold cross-validation as a balance between variance and computational cost: each fold holds ~ 124 examples for validation, yielding stable loss estimates without excessive runtime. A sweep over learning rates $\{1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}, 5 \times 10^{-6}, 1 \times 10^{-6}\}$ with up to 10 epochs per fold (patience = 2) produced mean best validation losses:

Learning Rate (LR)	Validation Loss
1×10^{-4}	3.2444
5×10^{-5}	3.2401
1×10^{-5}	3.2922
5×10^{-6}	3.3262
1×10^{-6}	3.3798

Table 2: Validation loss for different learning rates during fine-tuning

The lowest loss at 5×10^{-5} confirms that this learning rate best balances convergence speed and stability.

2.5 Regularization Strategies

With $\text{LR} = 5 \times 10^{-5}$ fixed, we evaluated two common regularizers:

- **Weight decay** ($\ell_2 = 1 \times 10^{-5}$): mean CV loss = 3.2478.
- **SBERT dropout** increased to $p = 0.1$: mean CV loss = 3.2480.

Neither technique produced a statistically significant improvement, suggesting that the pretrained SBERT weights already provide sufficient capacity and robustness for our contrastive task without additional regularization.

2.6 Extended Training & Final Schedule

Unlike the initial learning–rate sweep (limited to ≤ 10 epochs per fold, patience $= 2$), analysis of the resulting early–stopping statistics showed that no fold required more than 16 epochs to converge. Consequently, we fixed an upper limit of 15 epochs with patience $= 3$ for the definitive 5-fold cross–validation run, keeping the best learning rate LR $= 5 \times 10^{-5}$ and disabling weight decay and dropout. The best validation loss obtained in each fold is summarised in Table 3; the mean of 3.1560 confirmed that this schedule achieves stable generalisation without unnecessary computation.

Fold	Best Val. Loss
1	3.1808
2	3.1971
3	3.1791
4	3.1579
5	3.0651
Mean	**3.1560**

Table 3: Per–Fold Best Validation Losses for the Extended Training Schedule.

We then retrained on the full dataset using these settings and saved the final checkpoint (Version 6) as `playlist_finetuned_model`. To contextualize the efficiency of our pipeline for this section, Table 4 summarizes the hardware used and approximate compute requirements for each stage of the system.

Stage	Hardware	Wall-clock time	Compute-hours (approximate)
Pre-compute song embeddings (SBERT + numeric fusion)	NVIDIA T4 GPU	4 min	0.07 GPU-h
Embedding with fine-tuned SBERT	NVIDIA T4 GPU	7 min	0.12 GPU-h
Numeric-branch MLP training	NVIDIA T4 GPU	< 1 min	0.02 GPU-h
Language-detection pass (fast 5-word)	NVIDIA T4 GPU	14 min	0.23 GPU-h
Heavy re-check (50-word, English rows)	NVIDIA T4 GPU	8 min	0.13 GPU-h
Fine-tuning fusion head	NVIDIA T4 GPU	10 min	0.17 GPU-h
Interactive inference / Gradio	CPU only	< 1 min	negligible

Table 4: Compute Resources and Approximate Runtimes for Each Pipeline Stage

3 Results

Synthetic Playlist Experiments on Base Model

To illustrate the differences between Base and improved models, we present results for a sample prompt: "2000s Spanish pop songs to dance for", and compare the songs generated. We evaluated how many of the recommended songs in each case actually matched the attributes "Spanish", "pop", and high "danceability", based on our main dataset. The table below summarizes the empirical results, demonstrating that the fine-tuning approach, not relying on filters, effectively improved the model's attribute alignment.

Song	Danceability	Pop	Spanish
Where Blue Begins	Y	Y	N
Morsian	Y	Y	N
There Is a Kingdom	N	N	N
Deeper Waters	N	N	N
Reach Beyond	N	N	N
Enough	N	N	N
Cold Blooded	Y	N	N
Tightrope	N	N	N
Keep Movin' On	Y	N	N
Split the Sea	Y	Y	N
Conversate	Y	N	N
Cacti For Clothes	Y	N	N
The Chase	Y	N	N
Total	8/13	3/13	0/13

Table 5: Base Model (First 13 Songs)

Song	Danceability	Pop	Spanish
Skin Tight (feat. Efya)	Y	Y	N
Cola Song (feat. J Balvin)	Y	Y	Y
Pour Me Water	Y	Y	N
Palea	N	Y	N
Ge Oss Mål	Y	Y	N
Chingalinga	Y	Y	Y
TOUCH	Y	Y	N
Ai Se Eu Te Pego – Live	Y	Y	Y
I'll Be Fine (feat. Fire Boy)	Y	Y	N
Mamacita	Y	Y	Y
La Cumbia de Sergio el Hablador	Y	Y	Y
I Like That	Y	Y	N
Mala Mía – Remix	Y	Y	Y
Total	12/13	13/13	6/13

Table 6: Base Model Improved with Synthetic Playlists

Base vs. Version 5

We also compared the performance of our model (Version 5) against the baseline with the prompt "roadtrip rock songs" and parameters including `alpha`=0.2, `beta`=0.8, `gamma`=0, and `delta`=0, a year range of 1980–2005, and no restrictions on popularity or artist popularity. We observed a clear improvement over the base model. Version 5 not only produced higher cosine similarity scores but also retrieved songs that better matched the intended mood and theme of the prompt. In qualitative listening, the results were more coherent and genre-appropriate, reflecting stronger alignment with user intent (see Tables 7 and 8 for a comparison).

Rank	Song Title	Artist	Score
1	*Bombo*	Adelén	0.124
2	*Format*	I-Octane	0.119
3	*Bevis*	BENNETT	0.117
4	*Not Coming Up For Air*	Nabiha	0.115
5	*Nattuggla*	LBSB	0.113
6	*Look At You*	Seth Ennis	0.112
7	*Heat-Wave*	Daniel T.	0.110
8	*I'm Still In Love*	Alton Ellis	0.110
9	*New Gal*	Beenie Man	0.110
10	*Self Control*	Bebe Rexha	0.110

Table 7: Top 10 Retrieved Songs by the Baseline for the Prompt "roadtrip rock songs"

Rank	Song Title	Artist	Score
1	*Best of You*	Foo Fighters	0.909
2	*Fascination Street*	The Cure	0.909
3	*Holiday - Faded Ending*	Green Day	0.909
4	*Everything Your Heart Desires*	Daryl Hall & John Oates	0.908
5	*Suicide Messiah*	Black Label Society	0.908
6	*Do You Want To*	Franz Ferdinand	0.908
7	*Don't Treat Me Bad*	Firehouse	0.908
8	*Hello It's Me*	Todd Rundgren	0.908
9	*Movin' on Up*	Primal Scream	0.908
10	*Married With Children*	Oasis	0.908

Table 8: Top 10 Retrieved Songs by Version 5 for the Prompt "roadtrip rock songs"

3.1 User Preference Evaluation

Since playlist quality is ultimately subjective, we conducted a user preference evaluation with 57 participants. Each participant selected their own prompt and specified their desired weights for features such as semantics, popularity, genre, and language. For each prompt, two anonymized 20-song playlists were generated—one using our baseline model, and the other using our final model. Participants were then asked to indicate which playlist they preferred. Of the 57 responses, 48 ($\approx 84\%$) favored the final version playlist, while only 9 ($\approx 16\%$) preferred the baseline. This strong majority supports our quantitative results: users consistently found the fine-tuned, feature-aware playlists more relevant and enjoyable than those generated by the baseline system.

Model	Participants (n)	Preference (%)
Baseline	9	15.8
Final Version	48	84.2

Table 9: User Preferences

4 Limitations

Our system has several limitations, particularly when handling complex or situational prompts such as "roadtrip songs" or "songs for studying." These prompts often require nuanced, context-specific

interpretation, but the sentence embedding model we use tends to underweight such semantics. This is partly due to the limited representational capacity of the model and partly because the prompts themselves may lack sufficient keywords for accurately capturing intent. We chose a SentenceTransformer over a LLM to support fast, scalable nearest-neighbor retrieval across 90,000+ pre-embedded tracks, using an LLM at inference time would be prohibitively slow and costly.

Although we attempted to address this with contrastive fine-tuning on a set of prompt–playlist pairs from Spotify, the training dataset was relatively small and lacked diversity, particularly in situational and mood-based playlists. As a result, generalization remains limited, and improving performance would require access to a broader, more varied set of curated examples as well as additional training resources.

The model also struggles with fine-grained distinctions in musical characteristics. It was not explicitly trained to capture concrete audio properties like BPM, nor to reliably differentiate closely related genres (e.g., rock vs. metal) or languages (e.g., Portuguese vs. Spanish) based solely on embedding similarity. To compensate, we implemented explicit genre and language filters before the final ranking step, for instance, tracks that don't match a user's request for Spanish rock are excluded from the candidate pool. While this improves precision, it can also remove relevant songs due to metadata gaps or misclassification.

Language detection, in particular, presents further challenges. We classify language using only the first five words of a track's lyrics with a lightweight model to ensure speed, but this method can misidentify multilingual, non-lyrical, or instrumental songs. When combined with the strict filtering mentioned above, such misclassifications can lead to the exclusion of appropriate results.

Finally, the dataset itself introduces several limitations. Genre labels in the SpotGen dataset can be noisy or inconsistent, and some tracks lack lyrics altogether. Audio features such as danceability or acousticness are also subjective–what feels danceable to one listener may not to another—introducing variability into retrieval quality. Moreover, popularity measures are biased toward listening behavior from 2019, meaning older or legacy artists like Led Zeppelin may be underrepresented compared to more recent artists like Lil Uzi Vert, regardless of cultural influence.

Ultimately, our design decisions, including model selection, filtering, and reliance on metadata, were shaped by limited compute resources. However, despite these constraints, we show that natural language-based playlist generation is not only possible but practical with a content-based approach, laying the groundwork for richer, more robust systems in the future.

5 Conclusion

This work presents a fully modular, interpretable system for content-based music recommendation from natural language prompts. Rather than relying on historical user-item interactions, our approach integrates diverse metadata—lyrics, genre, language, and audio features—into a unified retrieval pipeline. Our workflow begins by preprocessing the SpotGen dataset: merging multiple tables, applying language detection, normalizing acoustic features, and synthesizing descriptive text for each track. These descriptions combine raw lyrics with natural-language cues from audio and categorical metadata, enabling richer semantic understanding.

We encode user prompts and songs into a shared latent space using a multimodal embedding architecture that fuses a SentenceTransformer-based text branch with an MLP-projected numeric feature branch. This embedding forms the basis of a multi-stage retrieval pipeline that incrementally introduces genre filtering, release year and popularity filters, language constraints, and audio feature boosting. Each version adds flexibility and alignment with user intent, while retaining transparency through explicit control parameters (α, β, γ, δ).

Comparing the qualities and features of songs retrieved by the baseline and final models shows a clear improvement in how well the results align with user intent. Empirical validation confirms this: in a user study with 57 participants, 84% preferred playlists generated by our feature-aware, fine-tuned system over those from the semantic-only baseline. Despite limitations stemming from dataset quality, model capacity, and hardware constraints, our results demonstrate that fast, scalable, and user-controllable playlist generation is achievable through content-based methods alone.

This system lays the groundwork for future research in prompt-based music retrieval. Future work could explore larger and more diverse datasets, improved model architectures including fine-tuned LLMs, more expressive embeddings, and generative techniques to support richer prompts, situational context, and adaptive user interfaces while maintaining interpretability. Additional directions include transfer learning from large music-focused models, deeper analysis of raw audio patterns (e.g., timbre, rhythm motifs), interactive feedback loops, and fine-tuned multimodal encoders to better capture nuanced listener preferences. These advances promise to enhance both personalization and retrieval depth without sacrificing transparency.

References

[1] van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, 26.

[2] Oramas, S., Nieto, O., Sordo, M., & Serra, X. (2017). A deep multimodal approach for cold-start music recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems* (pp. 32–37).

[3] Palumbo, E., Penha, G., Damianou, A., Redondo García, J. L., Heath, T. C., Wang, A., Bouchard, H., & Lalmas, M. (2025). Text2Tracks: Prompt-based music recommendation via generative retrieval. *arXiv preprint arXiv:2503.24193*.

[4] Shahane, S. (2022). Spotify and Genius Track Dataset (SpotGen). Retrieved from `https://www.kaggle.com/datasets/saurabhshahane/spotgen-music-dataset` (Accessed: 2025-05-02).

[5] Hugging Face. (2020). Sentence Transformers: all-MiniLM-L6-v2. Retrieved from `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2` (Accessed: 2025-05-02).

[6] Hugging Face. (2022). papluca/xlm-roberta-base-language-detection. Retrieved from `https://huggingface.co/papluca/xlm-roberta-base-language-detection` (Accessed: 2025-05-02).

Contributions

All members of the team contributed equally (25% each) across all stages of the project. This included dataset preprocessing, metadata engineering, embedding and model design, natural language sentence generation, retrieval architecture development, and evaluation. We collaborated closely during each model iteration, jointly implementing feature fusion strategies, tuning hyperparameters, and writing the final report and presentation. We each played an active role in both the technical implementation and conceptual refinement of the system, report, and slides.