PLAYLIST GENERATION WITH GPT2-MEDIUM

TECHNICAL REPORT

James Siegener[1], Trenton Slocum[1], Hildegard Younce[1]
[1]School of Data Science, University of Virginia, Charlottesville, VA, USA
gnq2mr@virginia.edu, nuf8ms@virginia.edu, ksg8xy@virginia.edu

April 28, 2025

ABSTRACT

In this project, we developed a Large Language Model (LLM)-based system to function as an AI-assisted playlist generator. Given a user prompt such as "Build me a study playlist with lo-fi and jazz influences", the model processes the request, retrieves relevant song data, and generates a curated playlist aligned with the user's preferences. This work was inspired by Spotify's "Energy Mix" feature, which curates playlists based on moods; however, we sought to extend this functionality to accommodate more nuanced and specific prompts. Our objective was to enable users to effortlessly create playlists tailored to any occasion, mood, or genre. To accomplish this, we leveraged data from the Spotify and Genius APIs to extract playlist features, metadata, and song lyrics. Our methodology utilized a GPT-2 Medium architecture capable of interpreting natural language queries and producing a corresponding list of song titles. Ultimately, this project aimed to streamline and enhance the playlist creation process by integrating the capabilities of LLMs with rich, publicly available music data, offering users a flexible and intuitive tool for music discovery and organization.

1 Introduction

In recent years, music streaming platforms like Spotify have made strides in personalizing the listening experience, offering curated playlists based on user activity and mood. However, users remain limited by the static nature of these playlists or must rely on manual search and filtering to discover music for specific themes or occasions. Motivated by the desire to create a more dynamic and interactive experience, we envisioned a system that allows users to generate personalized playlists through natural language prompts and receive not only song titles but also direct Spotify links. First, we focused on collecting a wide-ranging dataset of playlists and song metadata from Spotify, ensuring diversity across genres and time periods. Next, we gathered song lyrics from the Genius API with the goal of helping the model learn how musical themes, moods, and genres are expressed through language. We then trained a large language model to process user prompts and identify songs that align with the described criteria. Finally, we integrated the model into a user-facing web application that accepts text input and returns curated playlists. With additional time, we aimed to further personalize the system by incorporating a user's individual listening history into the training pipeline, allowing the model to better reflect their unique musical preferences.

2 Related Works

2.1 Song Recommendation with GPT-2

Recent work has explored using language models like GPT-2 and T5 to predict songs that best fit an existing playlist's mood, given a prefix of songs. These approaches leverage textual data such as song titles, lyrical keywords, and natural language representations of audio features to generate new song recommendations. While Spotify's recommendation system performs well for general user listening history, it struggles to suggest novel additions to custom playlists, often recommending songs the user already knows or by artists already in the list. In contrast, this study aimed to generate original tracks that match playlist characteristics, evaluated using BLEU scores and metadata classification accuracy across fields like danceability, energy, and valence. GPT-2 emerged as the most promising model, generating outputs

that were both intelligible and contextually aligned. While our project focuses on curating full playlists of existing tracks rather than generating original music, this work demonstrates the potential of using GPT-based models with Spotify data for intelligent, context-aware music recommendation.Chen and Teoh [2024]

2.2 Playlist Generation Using Reinforcement Learning

Other work has approached music playlist generation as a language modeling task, treating playlists as sequences of tokens and leveraging an attention-based RNN language model trained on baseline recommended playlists. By pretraining the model with this sequence data, researchers were able to optimize playlist generation using policy gradient reinforcement learning, tailoring recommendations to user profiles. This method offers a flexible, automated framework for generating personalized playlists in online streaming services. However, while effective, this approach does not incorporate natural language prompting, which limits user control and expressiveness—a gap our project addresses by enabling prompt-based playlist generation through large language models.Shih and Chi [2018]

2.3 OpenAI's GPT-2 Model

GPT-2, developed by OpenAI, is a transformer-based language model known for its strong performance in generating coherent and contextually relevant text. Trained on massive corpora, it excels at capturing semantic and syntactic patterns, making it highly effective for text-based generation tasks. Its architecture, based on self-attention mechanisms, allows it to model long-range dependencies and understand nuanced prompts. These capabilities make GPT-2 particularly appealing for our project, where interpreting user prompts and generating thematically consistent song lists benefit from natural language understanding. By framing playlist creation as a language generation task, GPT-2 provides a flexible foundation for prompt-based song recommendation systems. Radford et al. [2019]

3 Methodology

3.1 Data Collection

To collect data for our project, we used the Spotify API to gather information on playlists and songs. Although features such as retrieving the most popular playlists and accessing song-level attributes like "danceability" and "energy" had been deprecated, we were able to adapt by querying the top 50 playlists for a given keyword. We selected a wide range of keywords across different genres, themes, and time periods, collecting each playlist's name, description, and detailed song and artist information. This process was automated through a simple Python script to streamline API calls. Ultimately, we compiled a dataset of over 2,000 playlists. However, the reliance on keyword-based queries introduced bias, as the data was limited to playlists associated with the specific terms we generated rather than a random or fully representative sample.

In addition to collecting playlist and song information, we used the Genius API to scrape lyrics for over 99,000 songs in our dataset, automating this process through Python scripts. Our initial goal was to incorporate lyric data into our model to help it learn the thematic connections between songs and playlist descriptions. However, due to limited computational resources, reduced model performance, and time constraints, we ultimately decided to omit the lyrics data from our final analysis. We hope to explore the integration of lyrical information in future work.

3.2 Data Pre-Processing

The dataset used for training the model was derived from a collection of playlists, with entry containing the following columns.

- `Playlist_Name`: The name of the playlist
- `Playlist_Description`: A brief description or context of the playlist
- `Playlist_Songs`: A list of song names paired with their corresponding artist names

The dataset was first preprocessed to ensure that song-artist pairs in the `Playlist_Songs` column were converted to a consistent format, enabling easier tokenization during training. The `Playlist_Songs` column, which contained lists of song-artist pairs, was transformed into a string format where each pair was represented as "Song Name - Artist Name".

3.3 Model Selection

For the playlist generation task, we employed the GPT-2 Medium model due to its ability to handle long-range dependencies and generate coherent text based on prompts. GPT-2 is a causal language model that predicts the next word (or token) in a sequence, making it suitable for text generation tasks.

We selected GPT-2 Medium because of its balance between model size and computational requirements. It has 345 million parameters, providing sufficient capacity to generate high-quality outputs while being more efficient than the larger GPT-2 variants.

3.4 Fine-Tuning Process

To fine-tune the GPT-2 model, we used the Hugging Face Transformers library, which provides easy-to-use tools for training language models. The model was trained on the preprocessed dataset, where the input text for each sample consisted of the playlist name, description, and the list of songs paired with their respective artists. The output was a list of songs and their artists.

Training was done using the Trainer API, which simplifies the training loop by handling gradient accumulation, evaluation, and checkpointing. The model's parameters were fine-tuned using cross-entropy loss, where the labels were the same as the input tokens, as the task is a language modeling problem.

We trained our model for 20 epochs to allow sufficient exposure to the dataset and learn the mappings between playlist names and song names. We used a batch size of 2 due to the memory constraints of training with larger models on GPUs. A learning rate of 2e-5 was chosen as it provided a good balance between convergence speed and stability. The model also used mixed-precision training to optimize memory usage and computational efficiency.

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning method that inserts trainable low-rank matrices into the transformer layers, allowing adaptation with fewer resources. We attempted to make use of LoRA for more efficient memory usage and training. This allows for faster fine-tuning without requiring substantial computation resources by training on a small percentage of the pretrained LLM parameters rather than retrain the entire model.

3.5 Front-End Development

Once we had a fine-tuned model for playlist generation, we developed a web application to make the model accessible to users. We used the Django Python framework to build a simple HTML-based interface, with Django views handling POST requests from form submissions. Users can input a natural language playlist prompt and click "Generate," at which point the model is called with the user's prompt. The model returns a list of songs and artists, which are parsed and matched to corresponding entries in a DataFrame. The song name, artist, and Spotify URL are then displayed in a table. To enhance usability, the application caches previous prompts in server memory, enabling quick redisplay without additional API calls. For now, the application is deployed locally. An example output is shown in Figure 1.

4 Experiments and Results

4.1 Evaluation Based on Validation Loss

The first evaluation criterion used during model training is validation loss, which serves as a primary indicator of the model's ability to generalize to unseen data. The model's performance was monitored closely over the course of training, with checkpoints saved periodically to allow for comparison.

We used causal language model loss to assess the model's ability to predict the next token in a sequence. This concept uses the cross-entropy loss function to measure how well the model predicts the next word based on the preceding context.

Model selection was based on the lowest validation loss, ensuring the chosen model was not overfitting to the training data and retained the ability to generalize well to new, unseen inputs.

As seen in the table above, we trained two different models to test how differences in hyperparameter tuning might affect model performance.

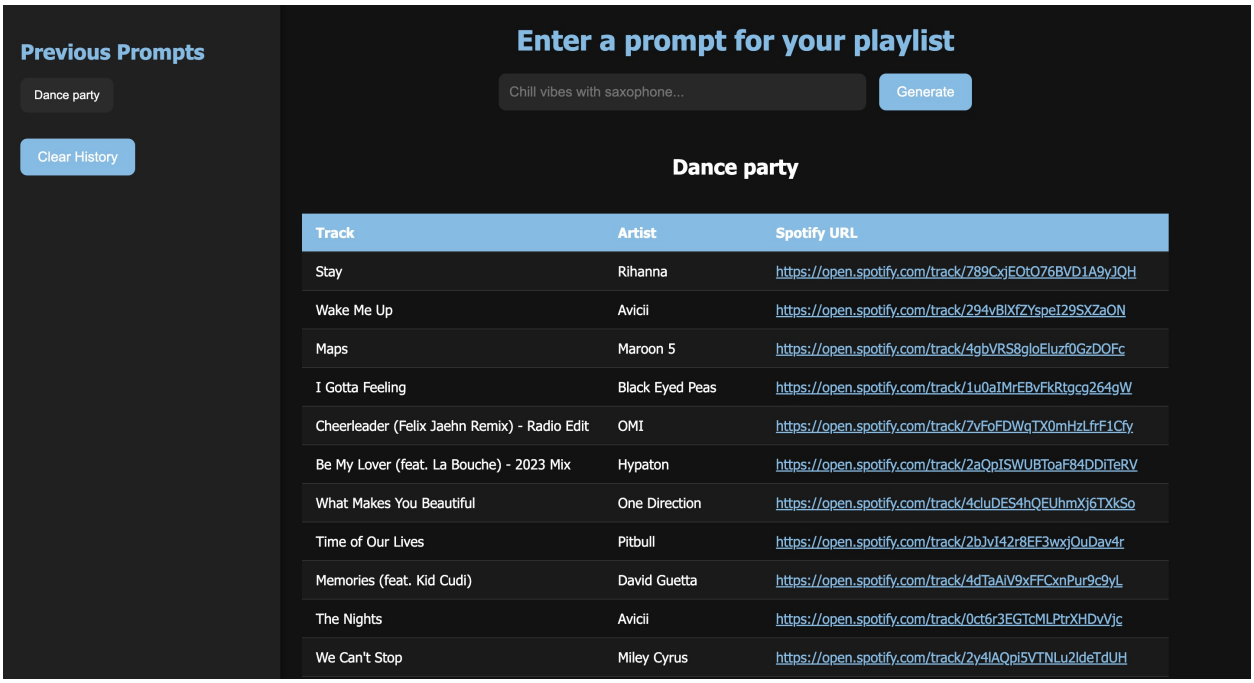

Figure 1: Example Output from Web Application

Table 1: Training and validation losses for standard vs. LoRA finetuning.

Model	**Training Loss**	**Validation Loss**
Standard Finetuned Model (345M params)	1.90	2.25
LoRA Finetuned Model (4.3M params, 1.2%)	1.89	2.27

4.2 Qualitative Evaluation: Human Approval

While quantitative metrics like validation loss are important for assessing model accuracy, our main goal was to generate playlists that humans would find meaningful and relevant. Thus, we conducted a qualitative evaluation where human assessors reviewed the model's output.

For this analysis, playlists were generated based on various input prompts, and the generated playlists were evaluated based on:

- Relevance: How well did the model match the genre, mood, and theme of the input prompt (e.g., "classic rock roadtrip")?
- Diversity: Did the playlist include a variety of songs and artists, ensuring a diverse listening experience?
- Creativity: How well did the model create unique and interesting combinations of songs and artists, especially when the input was open-ended or ambiguous?

Human evaluators were presented with several generated playlists and asked to rate each on a scale of 1 to 5, with 1 being "poor" and 5 being "excellent." The evaluators were instructed to focus primarily on the coherence of the playlist, its ability to fit the given prompt, and its potential usefulness in a real-world context (e.g., generating playlists for social sharing or personal use).

Overall, our model-generated playlists received mostly positive feedback. Many evaluators noted that the model was able to generate playlists that were quite relevant to the prompts and included popular, well-known songs and artists.

Some evaluators mentioned that while the model's output was generally coherent, it occasionally generated playlists that lacked the variety expected from human-curated playlists. In some cases, the same artist or song appeared multiple times within a single playlist, which impacted diversity. It was also mentioned that some prompts would not generate

related playlists. Despite these minor flaws, the overall human approval rate was high, indicating that the model's playlists were largely considered both relevant and enjoyable.

5 Discussion

The experiments yielded several important insights into the performance and trade-offs associated with different fine-tuning strategies.

First, when comparing the standard fine-tuned model and the LoRA fine-tuned model, we observed that both approaches achieved similar training and validation losses. The standard fine-tuned model achieved a validation loss of 2.25, while the LoRA fine-tuned model achieved a slightly higher validation loss of 2.27. Despite this small difference, the LoRA approach used significantly fewer trainable parameters—approximately 1.2% of the full model size—demonstrating its efficiency in terms of computational resource usage.

The minor increase in validation loss for the LoRA model suggests a modest trade-off in predictive accuracy for the sake of significantly reduced memory and computational overhead. Given the relatively small gap between the two models' performance metrics, LoRA fine-tuning appears to be an effective and practical alternative to full-parameter fine-tuning, especially when computational resources are limited.

However, it is important to note that loss values, while useful, do not capture the full picture of user satisfaction or practical usability. To address this limitation, we supplemented the loss-based evaluation with qualitative human assessment. This combined approach allows for a more comprehensive understanding of the model's effectiveness in the real-world use case of playlist generation.

6 Conclusion and Future Work

6.1 Conclusion

In this project, we developed a sophisticated playlist generation model alongside a simple yet professional web application to enhance usability. The model produced acceptable results based on qualitative assessments, although there is definite room for improvement. Our work demonstrates that with further development and greater computational resources, a tool like this could significantly improve the experience of creating playlists for users.

6.2 Future Work

There are several ways in which we could improve our model. The most immediately obvious shortcoming of our model was the bias introduced by our data collection. Since the only way to query playlists from the Spotify API was by using specific keywords, our model performed well when those keywords were used but struggled to generalize to unseen words. In the future, more aggressive scraping techniques could be employed to improve our dataset and reduce overfitting.

Another limitation was the difficulty matching the title of the songs with the correct artists. This issue could potentially be overcome by incorporating Retrieval-Augmented Generation (RAG) techniques to ensure that responses generate accurate song-artist pairs.

Finally, incorporating song lyrics into the model using enhanced computational resources could further improve performance. This would allow the model to learn patterns in the lyrical content and better match songs to natural language prompts.

7 Appendix A

All code and supplementary materials can be found at the following repository:

`https://github.com/jamessiegener/DS6051-Project`

References

Carrie Chen and Janice Teoh. Next-song recommendations for spotify playlists using gpt-2 and t5. Technical report, Stanford University, 2024. URL `https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1234/final-reports/final-report-169493994.pdf`. CS224N Final Project Report.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Shun-Yao Shih and Heng-Yu Chi. Automatic, personalized, and flexible playlist generation using reinforcement learning, 2018. URL https://arxiv.org/pdf/1809.04214. arXiv:1809.04214.