Semantic Search for Spotify with Sentiment

Nihong Yi, Sean Holland, Eric Tran, Felix Huang, Trung Le

# Problem

Currently, there are two ways to find songs using the Spotify music streaming service. The first is to manually use the search engine and get matches on keywords for song attributes such as track title, artist name, and albums. The second way is to scroll through curated playlists in the explore page that are categorized by things such as mood and music genre.

These constraints limit the exposure for some artists and restricts the users to refine their keyword search or rely on Spotify’s recommendation system to suggest tracks. One downstream effect of this is if a user is unable to find their song of choice efficiently, users are discouraged from using Spotify search and, by extension, Spotify itself.

# Solution

Our semantic search is meant to augment the existing search to not only return tracks based on keyword matching but also semantic similarity and lyric content. Unlike traditional keyword-based searches, semantic searching understands the meaning behind words and context, enabling more accurate and personalized results. The implications of this can be summarized in two pillars.

Firstly, for the artists, it will allow tracks with specific qualities to be discovered, thereby improving exposure for the artist. Secondly, streamers will be able to discover more relevant and refined music recommendations, ultimately improving the overall user experience on the platform. With semantic searching, Spotify can offer a more intuitive and intelligent search interface, helping users discover music that aligns more closely with their unique tastes and preferences.

The data set used in our solution is the [spotify-million-song-dataset](https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset). It includes attributes for the artist, song name, and lyrics. It has 57,650 songs with 643 unique artists. A snippet of the data is as follows:



By using semantic search, it puts less focus on remembering keywords. Users can search for meaning or general tone and have songs retrieved that match.

# Technical Implementation

In this section we will first walk through the process of our application, then talk about the technical details included in our solution.

## Technical Architecture

## 

Above is the diagram of how our application works. It is composed of offline and online modules. For the offline module, we preprocessed the data, created embeddings from the data, and indexed the embeddings with FAISS. More technical details will be discussed further in the report.

For the online module, every time a user inputs a query, we will use the query as a prompt and call the Llama 2 API, asking the model to generate keywords that are commonly used in lyrics and are related to the user’s query. We then use both the query and keywords to search through FAISS and retrieve the top results. To make the results actually playable, we call the Spotify API by passing the raw results and then create a playlist. This is the final result the user will see.

## Data Preprocessing

The first step in our approach involved extensive data preprocessing to prepare the dataset for effective analysis. This involved combining crucial textual data - specifically, the artist's name, song title, and lyrics. We focused on cleaning the data by removing special characters that could potentially skew the analysis. This step was crucial in ensuring the purity and usability of the data, laying a solid foundation for the subsequent stages of the algorithm.

## Embedding with All-MPNET

After preprocessing, we employed the all-MPNET model to generate embeddings. All-MPNET, a powerful language representation model, is adept at capturing the semantic essence of textual data. As we found in the model documentation, this specific model performed the best among several sentence transformer models in terms of sentence embedding performance and semantic search performance, with some tradeoff in terms of speed. By converting our textual data into vector space, we facilitated a more nuanced and computationally accessible representation of our songs. These embeddings serve as the backbone of our search algorithm, allowing for a sophisticated comparison of semantic similarities between songs.

## Indexing with FAISS using Hierarchical Navigable Small Worlds (HNSW)

The indexing of song embeddings was a pivotal step, for which we employed Meta AI Similarity Search (FAISS), specifically leveraging the Hierarchical Navigable Small Worlds (HNSW) algorithm. HNSW is an advanced method in nearest neighbor search, particularly effective in handling large-scale and high-dimensional datasets like ours. Its primary advantages include:

* **Speed and Efficiency**: HNSW significantly speeds up the search process, making it feasible to handle millions of songs without compromising on response time. This efficiency is critical for delivering real-time song recommendations.
* **Accuracy in High-Dimensional Space**: Given the complexity and high dimensionality of our song embeddings, HNSW offers a more accurate approach compared to traditional methods. It maintains high precision in similarity searches, ensuring that the most relevant songs are retrieved for a given query.
* **Scalability**: As our song database grows, HNSW's scalable nature allows for maintaining performance without a significant increase in computational resources. This is crucial for accommodating an expanding music library.
* **Proven Effectiveness**: Given its successful application in Spotify’s search algorithm, HNSW's use in our context provides a tried-and-tested solution, lending credibility and reliability to our approach.

By integrating FAISS with HNSW for indexing, we ensured that our music search algorithm could rapidly and accurately process user queries, retrieving the most relevant songs from a vast dataset with minimal latency. This approach not only improves user experience but also enhances the overall effectiveness of the search algorithm.

## Development of the Search Method

We developed a search method function that inputs a user query and returns a DataFrame of recommended songs. This function is the direct interface through which users interact with our algorithm. It processes the query, utilizes the pre-trained embeddings and FAISS index to retrieve relevant songs, and presents the results in an accessible format.

## Integration of Llama2 for Enhanced Query Interpretation

In an innovative twist, we experimented with incorporating Llama2, another advanced language model, to enhance query interpretation. Llama2 was employed to generate a list of keywords based on the user's query. These keywords were then used to calculate similarity scores against our song embeddings. This dual-model approach aimed to capture a deeper understanding of the user's intent and preferences from a sentiment standpoint, leading to more accurate and tailored song recommendations.

# Evaluation

## Offline Metrics

For offline metrics, we need to provide a very specific actual result list. Therefore, we cannot use queries that are too general and non-specific, as there would be a wide range of songs falling into this bucket, and the interpretation of “sad songs” can vary among people. So here, we generated queries with “hard” requirements that can lead to exact matches: in particular, either specifying the artist’s name or the actual lyrics. The two queries we used are as follows:

* Query 1: “songs by ABBA” (match the artist)
* Query 2: “songs with lyrics "Is this the real life? Is this just fantasy? Caught in a landslide No escape from reality Open your eyes Look up to the skies and see I'm just a poor boy, I need no sympathy"”

We used three metrics to evaluate the results: Recall@K, MRR (Mean Reciprocal Rank), and MAP (Mean Average Precision).

For Recall@K, we set K from 1 to 10 and check the recall respectively. We can see that for query 1, as we have a large number of actual results (113), the recall ratio is lower than 0.1 for top 10 results. However, it increases almost all the time as we increase K, meaning that it captures the songs by ABBA pretty well. For query 2, we only have one song in the actual result, and we returned the song at K = 7, meaning that we are able to match the query based on exact lyrics. Although we added keywords generated from Lllama as extra input, the rank of the song in the returned results was not high enough.

| K | Query 1 | Query 2 |
| --- | --- | --- |
| 1 | 0.01 | 0 |
| 2 | 0.02 | 0 |
| 3 | 0.03 | 0 |
| 4 | 0.04 | 0 |
| 5 | 0.04 | 0 |
| 6 | 0.05 | 0 |
| 7 | 0.06 | 1 |
| 8 | 0.07 | 1 |
| 9 | 0.08 | 1 |
| 10 | 0.09 | 1 |

For MRR, we set K = 10 and take a look at the reciprocal rank for each query, and the mean reciprocal rank for two queries. We can see that for query 1, where we are trying to find songs by ABBA, our RR is as high as 1, this means that we can hit the result on top 1 position. For query 2, where we are matching the lyrics, the RR is 0.14, leading the MRR to 0.57. The result shows that we can accurately retrieve the results from the input, even if the input is supposed to be very strict and specific.

* RR for first query: 1.0
* RR for second query: 0.14
* MRR: 0.57

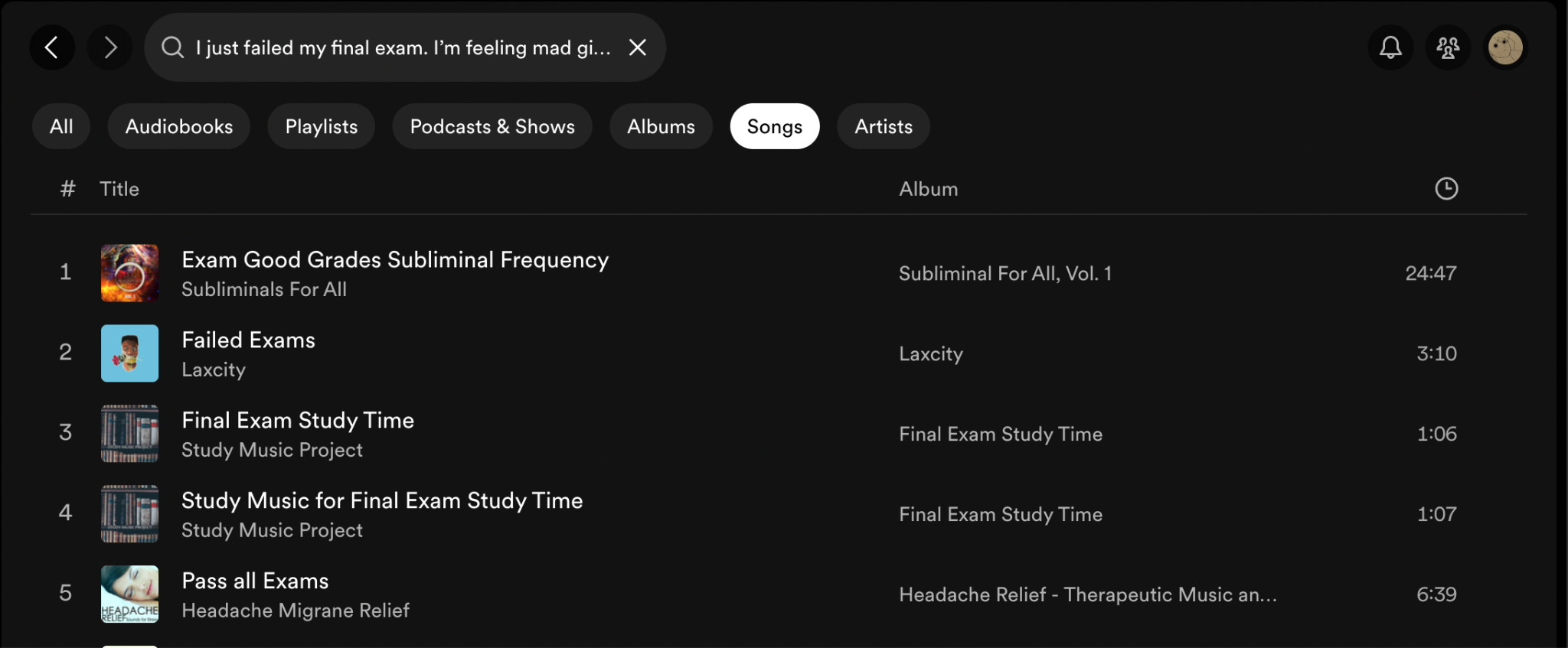
Similarly, for MAP, we can see that for query 2, the AP (average precision) is 0.14, which is the same as RR since we only have one actual result; on the other hand, for query 1, because of the large denominator from actual results, the AP of 0.09 is already pretty accurate. The MAP in this case seems to be low, but in reality it is performing very well.

* AP@10\_query1 = 0.09
* AP@10\_query2 = 0.14
* MAP@10 = 0.12

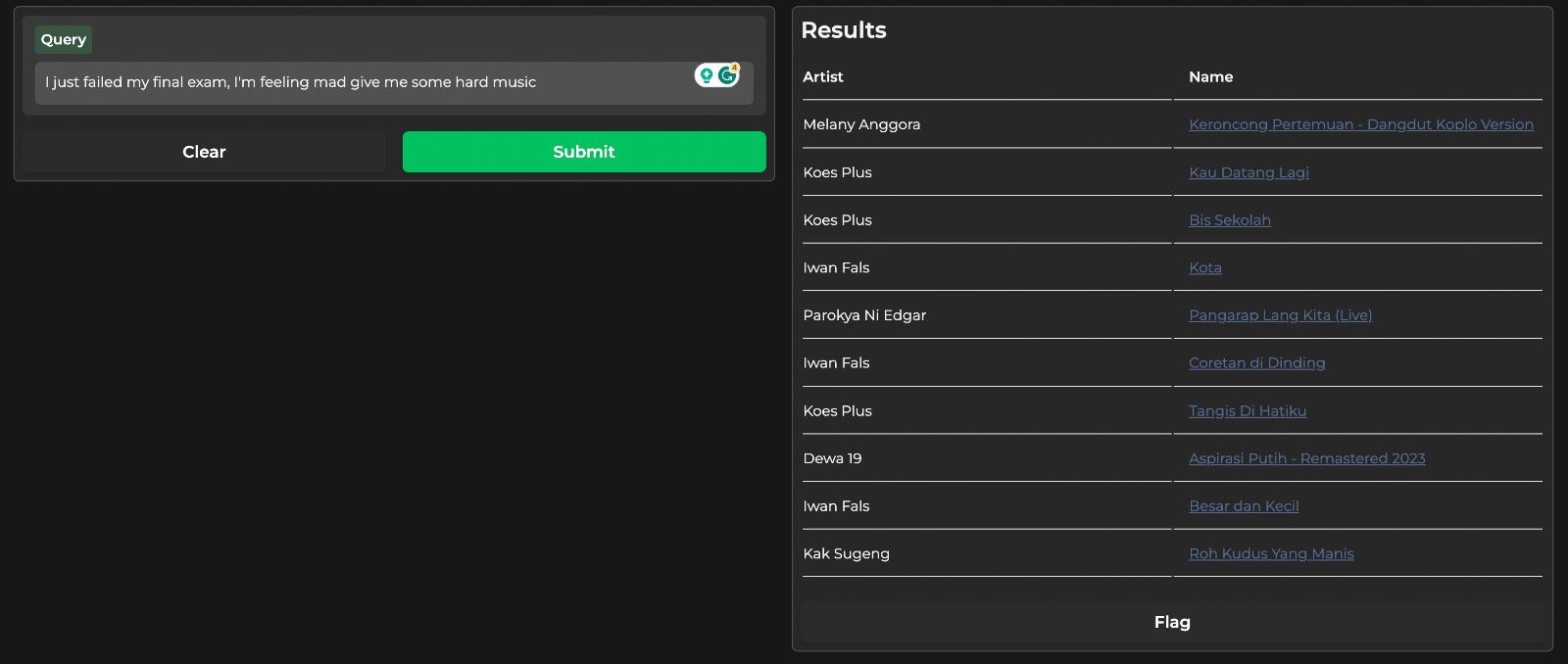
## Qualitative Evaluation

To test the performance of general queries, we compared the results we got from Spotify’s current search function and our application and used our judgment to evaluate the results.

For queries with semantic requirements, the current Spotify search is only able to return results with keywords that match the query:



The semantic search is able to return more accurate and personalized results by understanding the meaning behind words and context:



# Future Considerations

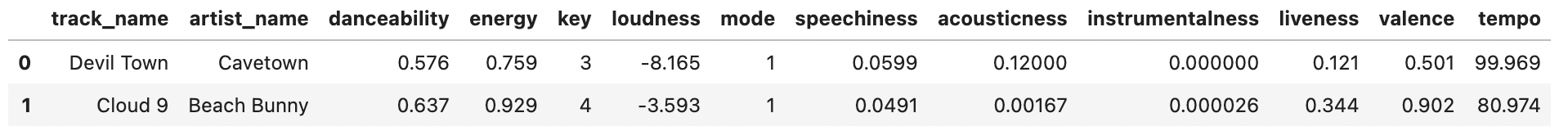
## Implementing new dimensions into the model

While we believe our new search will greatly enhance the Spotify experience for many parties, we also believe we could improve our project. By focusing on sentiment over keyword search we already added a new element to how listeners explore and engage with music. However, this could be improved by adding more elements of the track:

* **Beats per minute:** Lyrics are greatly affected by the music they are played with, this metric could dramatically improve our search.
* **Song era:** Quantifying when and where songs were written allows us to cluster based on genre or style, look at examples such as Southern Blues from the 1920’s and grunge from the Pacific Northwest from the 1990’s.
* **Tone/Vocal style:** Whether it is powerful opera, raspy rock, or anything in between, the vocal style plays a huge role in how people connect to a song.
* **Valence:** The scored ‘happiness’ of a song
* **Energy:** The scored energy levels of a song

These are all factors that go into the listening experience. If we could better implement these into our model, this would expand our prompts to include things such as “provide me with some fast paced songs about regret”, “give me songs that sound like they are from the 70’s”, or “list songs that have soulful vocals”.

For example, when querying the Spotify API for audio tracks, we are able to retrieve certain audio qualities of the song:



We can then use these qualities to infer certain qualities of a track to further augment our embeddings. For example, tracks that have high valence scores and are energetic can be labeled as Happy/Joyful. We can use this framework to improve our embeddings to include the musical qualities of the track.

## 

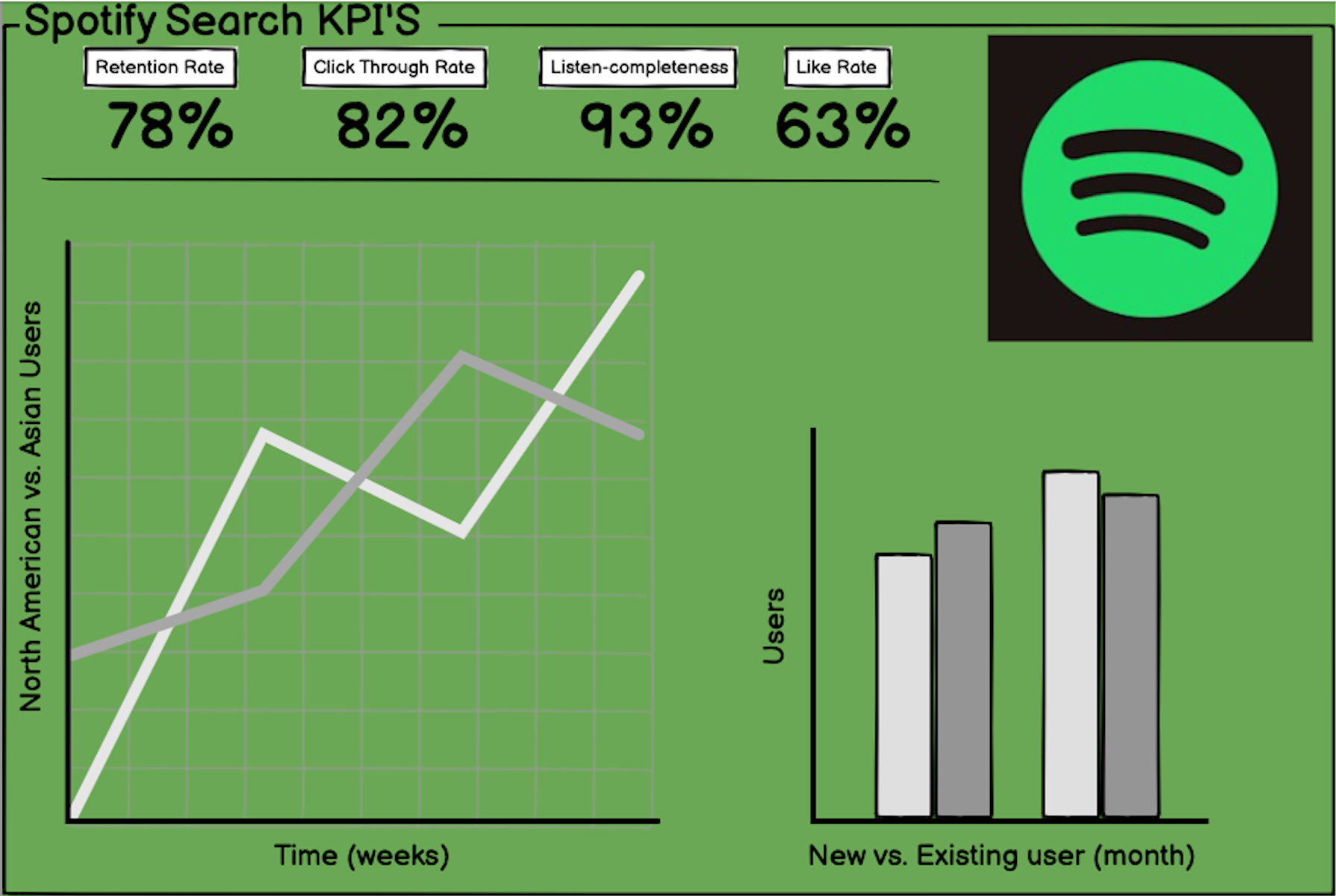
## AI DJ

Another potential area of growth for our project would be incorporating our prompt with the current “AI” Spotify DJ. Using voice recognition with our current model, these prompts would be faster and more personalized for the user. This would add to the interactivity of the entire user experience, which anecdotally is a complaint we have noticed about the current Spotify DJ.

## KPIs

To improve our model as it is being used, we would create and measure some KPI’s. Since we are recommending music based on the users prompt, all of our metrics should be ways to determine how much the users liked our suggestions. Our metrics would be:

* **Click through Rate:** Out of the songs we recommend, how many are listened to by the user.
* **Listen-completeness-rate:** For each song we recommend and the user plays, what % of the song do they complete. A high completion rate indicates that the songs not only match the initial search prompt but also sustain the user’s interest throughout the track. On the other hand, a lower rate may signal a need for further refinement in the algorithm’s selection criteria or understanding of intent.
* **Like-rate:** Does the user add our recommendations to their liked song, is the recommendation the reason they know of the song
* **Retention:** how many users come back for the feature again/ how many users have tried the new feature



These KPIs can be tracked in an accessible dashboard (such as in the example above) for relevant senior management.

# Reflection and Discussion on Overall Learning

This project served as a great synthesis of what we learned from this course, particularly with regards to the specific technical implementations related to large language model concepts (embeddings, vector databases, semantic search) but also with lessons throughout the duration of our MSBA curriculum, including but not limited to: proper code development, presentation development, teamwork and communication skills, and professionalism in an end-to-end data science project.

Our team faced a number of challenges throughout this project. For example, we experimented with new and different packages, libraries, and models not explicitly covered in the course - to successfully incorporate these new items in our project in a way that we could showcase our growth, we had to take what we had already learned in the course as background, general context and adapt to new circumstances (and read plenty of documentation).

We also were challenged to connect our key findings and takeaways to business impacts. Although we started with some good ideas on the direction that we wanted this project to go and the overarching motivations, we deeply considered the user, the user experience, and how our new app solved user pain points. In some ways, this process drew upon our learnings from other courses in the MSBA curriculum such as product management, but ultimately our group felt that this also showcased our growth as not just pure data scientists but as business analysts who can take real business problems, turn them into data analytics projects, and come out with actionable takeaways and summaries.

As mentioned previously, since our team incorporated previously unseen tools, we naturally were excited by certain components of our work by the end of the project. Notably, our team enjoyed working with the Spotify API and the depth of Spotify data available. Granted, we were unable to work with music audio-related data (and mentioned in our presentation and report that we would like to do so in a hypothetical future iteration of this project), but the audio-related data that we were able to find, in addition to the lyrics data that we had to begin with, were interesting to work with. Our team also liked working with the Llama 2 model for keyword generation. It was fun to be able to try out a different generative model and incorporate it in a useful way. Lastly, our project group enjoyed interacting with the front-end Gradio interface and developing example queries. Especially as all of us are avid users of Spotify, it was interesting to develop our own (basic) app interface, still with the recognizable stylings of Spotify, and experiment with different search terms across both Spotify itself as well as our deployed app and analyze the difference in results.

Ultimately, this project was a success end-to-end. We were pleased with the outputs (code, app, report, and slide deck) and enjoyed working on a project with topics that we were all interested in and passionate about.