CSE6242: Final Report

SoundSensei: Customizable Playlist Curation and Safe Listening for All

Team Data Detectives: Arya Mohan, Jake Michiels, Koushika Kesavan, Maitreyee Talnikar, Rajvi Parekh

1 Introduction

SoundSensei, an innovative music recommendation system, transcends the conventional boundaries of music curation by placing transparency, customization, and control at its core. Crafted for both passionate music enthusiasts and discerning artists, our platform goes beyond mere playlist analysis, offering users the power to tailor recommendations according to their unique preferences. With intuitive controls and customizable parameters, we ensure a user-centric musical journey that resonates with individual tastes.

What sets SoundSensei apart is its unwavering commitment to safety, providing parents with the tools to curate kid-friendly content seamlessly. Our target audience spans avid music listeners and those eager to delve deeper into the nuanced tones of content. The successful implementation of SoundSensei promises a significant impact on the landscape of comprehensive music analysis and personalized recommendations, a promise we aim to validate through rigorous professional user studies and continuous user feedback. Embark on a musical odyssey with SoundSensei, where every note is tailored to your distinctive rhythm.

2 Problem Definition

SoundSensei is a sophisticated music recommendation system centered on transparency, customization, and control. Tailored for music enthusiasts and artists, our system not only analyzes your playlist for insights but also provides customizable parameters for personalized recommendations. Users can refine playlists through intuitive controls, ensuring a user-centric musical journey. Safety is paramount, empowering parents to curate kid-friendly content. Our target audience includes avid music listeners and those seeking a better understanding of content tones. Successful implementation promises a substantial impact on comprehensive music analysis and customizable recommendations, gauged through professional user studies and feedback.

3 Literature Survey

Music recommendation systems are broadly categorized into two types: collaborative filtering and content-based recommendation. Collaborative filtering (Ricci et al.) relies on user history, often using latent factor models. However, it struggles with items lacking usage data. Content-based recommendation methods analyze item features. In the context of music, a model like (Van den Oord et al.) suggests songs perceptually similar to a user's past choices, using basic audio features or deep learning.
In music recommendation, "Automatic Playlist Continuation" aims to create evolving playlists. Ferraro et al. suggests a hybrid system using Matrix Factorization and track co-occurrences, but it may miss some relevant tracks. Kelen et al. offers a lightweight playlist-based nearest neighbor approach, optimizing computational usage at the cost of accuracy. Yang et al. addresses popularity bias and cold-start problems, relying on playlist titles, which could affect accuracy. Volkovs et al. combines collaborative filtering and deep learning but lacks user configurability, relying on past listening patterns.

In the music recommendation domain, ensuring kid-friendly content is a significant challenge. For instance, Zhang et al. studies user behavior and aligns with our goal to empower users and provide parental controls. Moreover, Schwind et al. introduces a QoE tool for Spotify, understanding network and app impacts on user experience, with some QoE estimation limitations. Similarly, Anderson et al. explores Spotify's user experience, emphasizing diversity and algorithmic recommendations. To address these limitations, we consider a broader user base and align with our project's playlist continuation and parental control features.

Sentiment analysis enhances recommendation accuracy and ensures content filtering for younger audiences. Studies like Gao et al. use techniques like word extraction and cosine similarity. Sarin et al. emphasizes context, emotion, and mood-based recommendations with Tableau visualization, while Rosa et al. refines sentiment analysis through a lexicon-based intensity metric. Integrating sentiment from music (lyrics/audio) and musical reviews can improve sentiment analysis and recommendations.

Sentiment analysis is vital in natural language processing, enhancing recommendation accuracy, especially for younger audiences. Gao et al. employs word extraction and cosine similarity in music recommendations. Sarin

et al. focuses on context, emotion, and mood-based music recommendations with Tableau visualization. Rosa et al. uses a lexicon-based sentiment intensity metric, considering factors like age, gender, and sentiment word tense, benefiting our project. However, integrating sentiment from music (lyrics/audio) and musical reviews is an untapped potential for enhanced sentiment analysis and music recommendations.

4 Proposed Methodology and Innovations

4.1 Outline

Our project leverages an integrated approach, focusing on automatic playlist continuation and playlist summarization using data from songs and their lyrics, bringing forth unique innovations. Here's a detailed outline:

1. Data Collection and Exploratory Data Analysis (EDA): We employ the Spotify API for acquiring song data and the Genius API for lyrics data, laying the groundwork for our analysis. We performed extensive EDA to gain a deep understanding of the dataset, which is crucial for effective modeling.

2. Playlist Summarization: Our system features an innovative playlist summarization tool. This includes a thorough EDA of user playlists, focusing on audio features and sentiment analysis via IBM Cloud's Natural Language Understanding. We also incorporate vulgarity analysis using the Perspective API. These steps help users understand their musical tastes. The summarization provides insights into the playlists' overall mood and themes, thereby enhancing user engagement.

3. Automatic Playlist Continuation: Our automatic playlist continuation model uses an autoencoder architecture in conjunction with cosine similarity to generate song recommendations that complement the user's existing playlist. We provide controllability and transparency in our recommendations, allowing the user to control different audio features they want in their recommendations and providing insight into why they were recommended particular songs. Another notable addition is the 'kid-friendly' mode, which filters explicit lyrics, making the playlists appropriate for younger listeners. This system is designed to deliver dynamic, customizable personalized recommendations.

Our project stands out for its user-centric design and enhances traditional music recommendation systems by innovatively integrating sentiment analysis, vulgarity detection, and an advanced autoencoder-based model for playlist continuation. Additionally, our system equips users to control the recommendation process through various input factors, a feature often lacking in contemporary recommender systems. The system's configurability and the inclusion of a kid-friendly mode cater to a diverse audience, ensuring safety and an enriching experience for all users, including children. This approach meets the evolving demands of modern listeners, making our system a significant step forward in personalized music experiences.

4.2 Data Collection and Pre-Processing

Our project utilizes several Python packages to collect and process the data, including BeautifulSoup, Spotipy, GeniusLyrics, and spaCy. Key steps in the data collection pipeline include:

1. Web scraping the Wikipedia page for the Billboard Year-End Top 100 charts for each year to collect the song name, artist name, and chart position. We collected data from 1960 to 2022.

2. Using the Spotipy library to access the Spotify API and collect audio features for each song, like danceability, energy, acousticness, duration, explicit content, etc. Special handling of errors was required as some songs did not have exact name matches in Spotify's database. Fuzzy name matching techniques were applied to find the closest match when an exact one was not found.

3. Accessing the Genius Lyrics API via the lyricsgenius package to collect lyrics for each song and passing it to Perceptive API to identify different types of explicit content such as obscenity, toxicity, or violence.

We then preprocessed the data by cleaning null values, converting lists to proper datatypes, one-hot encoding categorical features like artist name and genre.

The final preprocessed dataset contained close to 6000 rows representing songs, with columns for audio features, artists and genres, release year, duration, explicit content, etc. This dataset was then used to perform exploratory data analysis and train machine learning models to understand similarities between songs.

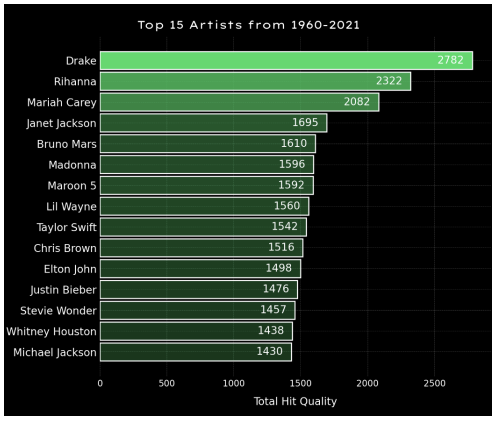
4.3 Exploratory Data Analysis

We performed extensive data analysis to gain insights into music trends and attributes. We created several vi-

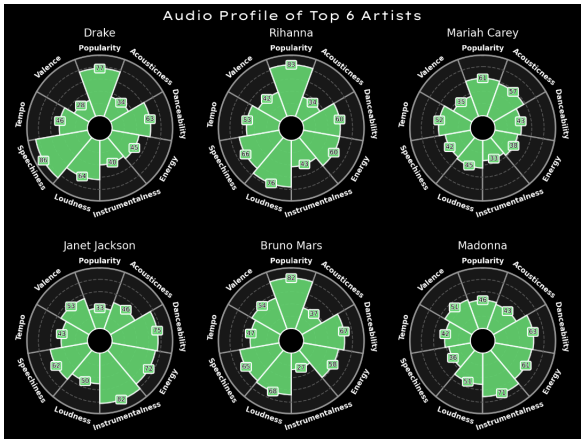
sualizations done systematically across four facets - audio features, artists, genres, and songs. The analysis answers key questions like:

- How have characteristics of popular music changed over time? Plots of audio features by year show increasing danceability, decreasing instrumentalness, more aggressive vocal styles etc.
- Who are the most successful artists and genres over the 60 year period? Hit quality metrics combined with horizontal bar plots reveal the top musicians across different eras.
- What are the distinctive audio profiles of top artists? Pizza charts visualize the percentile ranks of famous artists like The Beatles, Michael Jackson, Taylor Swift and others.
- How do artists cluster based on the genres they span? Network graphs illustrate the connections between artists and genres.
- How prevalent are different types of explicit content in popular music across different eras and genres? Through vulgarity analysis using sentiment and language processing tools, we examine trends in explicit content, identifying shifts in the usage of explicit lyrics over time and across various music genres.

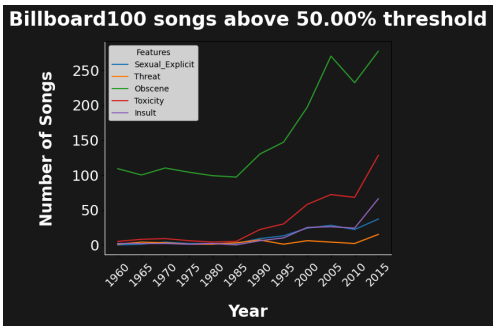
These visualizations help bring the underlying data to life beyond just training models. Figure 1 shows a few visualizations that we gained interesting insights from.



(a) Top 15 Artists from 1960-2021



(b) Audio Profile of Top 6 Artists



(c) Vulgarity analysis across the years

Figure 1: Examples of EDA

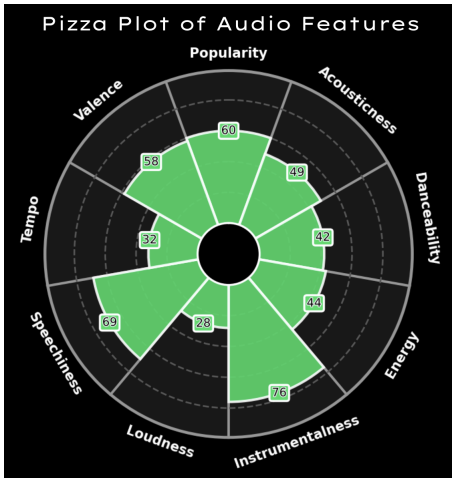
4.4 Playlist Summarization

We performed provide a comprehensive summary of user-uploaded playlists using visualization and other data analytics techniques. Key features such as audio characteristics are visualized as seen in Figure 2. Lyrical sentiment was also assessed using IBM Cloud's Natural Language Understanding, offering insights into the emotional tone and thematic content of the playlists. We also evaluated the presence of explicit content in lyrics, ensuring the suitability of playlists for diverse audiences. This approach allowed us to generate detailed summaries of playlists, providing users with a deeper understanding of their music preferences, the mood and themes present in their playlists, and enabling a more personalized listening experience.

4.5 Automatic Playlist Continuation

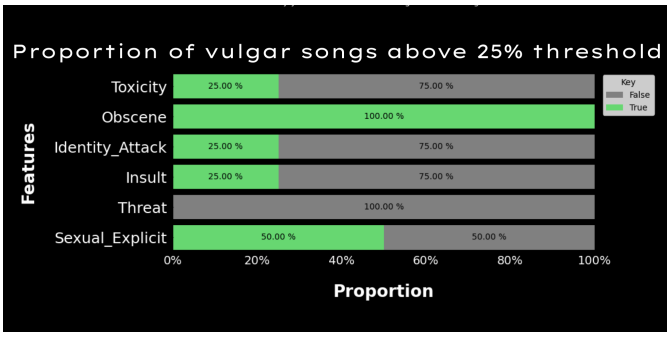
We developed a recommendation system comprised of three key elements: sentiment analysis, vulgarity analysis, and an automatic playlist continuation recommender. Each of these components are explained in detail below.

First, our sentiment analysis model focused on giving scores to each emotion category ("joy", "sadness", "anger", "fear" or "disgust") using song lyrics. We leveraged the IBM Cloud's Natural Language Understanding API for this task. We fine tuned this pre-trained model and normalized the scores for each emotion to put it in a scale of 0 to 1.



(a) Pizza Plot of User's Audio Features

(b) User's Audio Aura based on Sentiment



(c) Proportion of User's Songs with Vulgar Lyrics

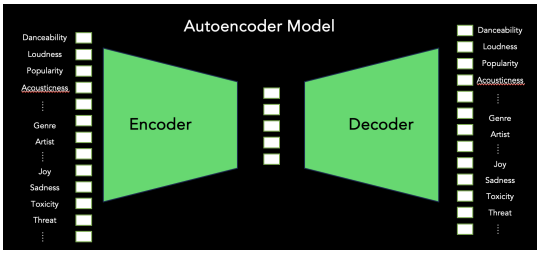
Figure 2: Playlist Summarization

For vulgarity detection, we used Perspective API to identify different types of explicit content within the lyrics. In addition to text toxicity, Perspective API provides scores for insult, profanity, identity attack, threat, sexually explicit content on a scale from 0 to 1 representing the probability that that type of content is present. This gives the user more insight into the reasons a particular song might be classified as explicit.

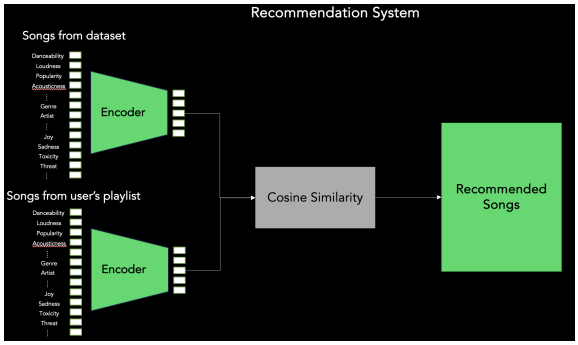
Next, we integrated our sentiment and vulgarity models into our automatic playlist continuation recommender system. Our recommender system uses an autoencoder neural network to create latent embeddings. The model learns a sparse vector embedding for each song. When a user inputs a set of preferred songs, embeddings for these songs are averaged and used to calculate cosine similarity with all other song embeddings. This method ensures that the top matches represent songs with similar latent features, and can be recommended.

Furthermore, our system enhances user autonomy by providing users the capability to customize music recommendations, including a 'kid-friendly mode'.This is achieved by allowing users to specify their preferred levels for various audio features like Acousticness, Danceability, Energy, and others. Our system then calculates the distance between each song's features and these user-defined values, ensuring recommendations align closely with personal preferences. Additionally, the incorporation of a profanity filter enables users to exclude songs with explicit content, making the platform versatile and suitable for a broad audience.

This combination of feature-based personalization and content filtering gives us a sophisticated recommendation algorithm that not only matches songs to the user's existing playlist but also adheres to their specific audio preferences and content sensitivities, enhancing the overall user experience.



(a) Autoencoder training



(b) How the songs are recommended

Figure 3: Automatic Playlist Continuation Model

4.6 Web Application

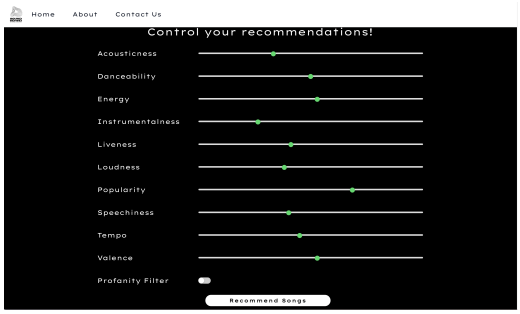
To ensure a smooth user experience, we designed a web application that combines the playlist summarization

and recommendation features previously described. This application was designed using wireframes in Figma and built using Flask for the backend and ReactJs for the frontend. The backend consisted of REST APIs responsible for managing tasks such as retrieving user playlists, creating analytics charts, and generating recommendations. The user interface was divided into several distinct sections, each asynchronously utilizing these APIs to perform different functions within the recommendation process. These sections include:

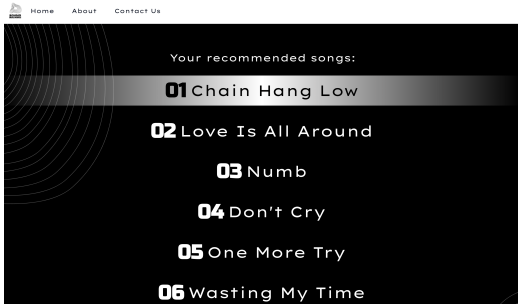
1. A home page that prompts users to retrieve their Spotify playlists.
2. An authentication page where users grant the app permissions to access their spotify data via an OAuth link
3. A playlist listing page for viewing and selecting playlists for analysis.
4. A song listing page showing songs from a selected playlist, with an option for detailed analysis.
5. An analytics page presenting playlist summaries, including sentiment, audio features, genre analyses, and a word cloud. It also has sliders for customizing recommendations and a toggle for filtering inappropriate content.
6. A recommendation section listing songs based on the user's selected playlist.

Few of these sections in the web app can be seen in Figure 4.

(a) Playlist Upload and Display



(b) Sliders to control recommendations



(c) Recommended Songs

Figure 4: Few pages of our web application

5 Evaluation and Observations

5.1 Results

The testbed consists of three primary models:

Sentiment Analysis Model: Utilizing the IBM Cloud Natural Language Understanding API, our sentiment analysis model achieved an accuracy of approximately 92% on a test set. This high level of accuracy, indicates the model's effectiveness in accurately classifying the emotional content of lyrics, which is crucial for the recommendation system's efficacy.

Vulgarity Model: This model plays a crucial role in identifying songs containing explicit content, flagging them for potential exclusion in our recommendations. To evaluate this model, a small set of songs from a variety of genres containing explicit or clean lyrics were labeled for each category. We found that the model is very effective at detecting obscene language. It consistently returns scores close to 1 when obscene language is present and scores close to 0 when not present. Using a threshold of 50%, the model correctly identified obscene songs 98% of the time. The model struggles to identify the other categories with high confidence and returns roughly .5 on explicit examples and roughly 0 on clean examples. This is likely due to the figurative nature of song lyrics. Fortunately, setting a low threshold ($\sim$25%) for these categories allows the model to classify songs correctly 89% of the time. Future work in this area should be focused on improving the confidence of highly figurative examples, and expanding the types of explicit content the model is able to detect (i.e. drug references, self-harm, gang violence, etc).

Automatic Playlist Continuation Model: The autoencoder exhibited a remarkably low mean squared

error (MSE) of ∼0.0052 in its loss function on the validation set. This low MSE indicates a high degree of accuracy in feature reconstruction, indicating that the embedding space can be reliably used for our recommendation system. We evaluated our final recommendation system using precision and recall metrics. It showed a balanced performance in providing relevant (high precision of ∼91%) and diverse (high recall of ∼87%) song suggestions. This balance enhances the user experience by aligning closely with user preferences and introducing new musical elements.

5.2 Experiments and Hyperparameter Tuning

Our experimental phase involved detailed hyperparameter tuning and yielded several key observations:

Autoencoder Architecture: We experimented with varying the number of hidden layers and neurons in the autoencoder. It was observed that increasing the depth of the model improved feature extraction but also led to a risk of overfitting. To mitigate this, we implemented dropout layers, which helped in regularizing the model.

Learning Rate Optimization: Adjusting the learning rate for the autoencoder was critical. A higher learning rate initially accelerated the convergence but often led to oscillations in loss. A gradual decrease in the learning rate, implemented via learning rate schedulers, provided more stability and a consistent decrease in the validation loss.

Distance Metric Selection: An essential aspect of our recommendation system was the selection of an appropriate distance metric for calculating similarities between songs. After experimenting with various metrics such as Euclidean, Manhattan, and Cosine similarity, we ultimately chose Cosine similarity for its effectiveness in high-dimensional spaces. This metric provided a more nuanced understanding of song similarity by comparing the orientation rather than the magnitude of vectors in the embedding space. This approach significantly improved the relevance of our recommendations, as it more accurately reflected the subtle similarities between songs in terms of their features and user preferences.

Profanity Filter Sensitivity: The profanity filter's sensitivity was fine-tuned to strike a balance between filtering explicit content and maintaining a diverse range of song recommendations. Higher sensitivity led to a more stringent filtering process, which, while beneficial for a 'kid-friendly' playlist, sometimes limited the variety of recommendations.

Distance Metric	Precision	Recall
Euclidean Distance	0.88	0.93
Manhattan Distance	0.73	0.65
Euclidean Distance	0.91	0.87

Table 1: Comparison of distance metrics

6 Conclusion

In summary, SoundSensei provides customizable and safe recommendations. Its core strengths lie in its ability to provide transparent and personalized music recommendations, coupled with robust safety features for family-friendly content. The utilization of advanced techniques, such as sentiment analysis and vulgarity detection, distinctly positions SoundSensei in the realm of music curation and recommendation systems.

Our system's ability to provide highly accurate sentiment and vulgarity analysis, coupled with a robust automatic playlist continuation model, sets a new standard in personalized music curation. The autoencoder's low MSE affirm the system's proficiency in understanding and processing complex musical data. Furthermore, the precision and recall metrics of our recommendation system highlight its effectiveness in not only aligning with user preferences but also introducing new and diverse musical elements, enhancing the overall user experience.

Looking forward, the potential for expansion and adaptation is vast. SoundSensei could evolve to encompass a broader range of music genres and languages, making it more inclusive and globally applicable. Incorporating user feedback loops and machine learning models that adapt to individual user preferences over time could further enhance the personalization aspect. Furthermore, the exploration of more advanced natural language processing techniques to further enhance sentiment and vulgarity analysis could also provide more nuanced and accurate recommendations.

In conclusion, SoundSensei offers a unique blend of personalization, user control, and safety, and sets a strong foundation for future innovations in this field.

Every team member has contributed a similar amount of effort.

References

A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas. Algorithmic effects on the diversity of consumption on spotify. In *Proceedings of The Web Conference 2020*, WWW '20, page 2155–2165, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380281.

A. Ferraro, D. Bogdanov, J. Yoon, K. Kim, and X. Serra. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–5. 2018.

J. Gao, H. Yuan, L. Wang, and Y. Qian. Evaluating sentiment similarity of songs based on social media data. In *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–6. IEEE, 2018.

D. M. Kelen, D. Berecz, F. Béres, and A. A. Benczúr. Efficient k-nn for playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–4. 2018.

F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2010.

R. L. Rosa, D. Z. Rodriguez, and G. Bressan. Music recommendation system based on user's sentiments extracted from social networks. *IEEE Transactions on Consumer Electronics*, 61(3):359–367, 2015.

E. Sarin, S. Vashishtha, S. Kaur, et al. Sentispotmusic: a music recommendation system based on sentiment analysis. In *2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)*, pages 373–378. IEEE, 2022.

A. Schwind, F. Wamser, T. Gensler, P. Tran-Gia, M. Seufert, and P. Casas. Streaming characteristics of spotify sessions. In *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2018. doi: 10.1109/QoMEX.2018.8463372.

A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

M. Volkovs, H. Rai, Z. Cheng, G. Wu, Y. Lu, and S. Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–6. 2018.

H. Yang, Y. Jeong, M. Choi, and J. Lee. Mmcf: Multimodal collaborative filtering for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–6. 2018.

B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, and D. Epema. Understanding user behavior in spotify. In *2013 Proceedings IEEE INFOCOM*, pages 220–224. IEEE, 2013.