UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science and Computational Engineering

Application of LLMs and Embeddings in Music Recommendation Systems

Abu Mohammad Taief

Master thesis in Applied Computer Science … May 2024

Abstract

Music recommendation systems are crucial in guiding listeners to the extensive selection of music accessible today. However, these recommendation algorithms frequently encounter challenges such as the cold start problem, including less popular tracks, and comprehending the semantic substance of music. This thesis investigates the incorporation of Large Language Models (LLMs) and embeddings to tackle these difficulties in the context of music recommendation systems. This project utilizes two extensive datasets from Kaggle to create a hybrid recommendation model that blends content-based and collaborative filtering approaches with proficient LLMs.

The main goal is to make music suggestions more accurate and customizable. To do this, two methods are being used, which include adding embeddings to classic collaborative filtering techniques and using LLMs to add semantic analysis to content-based recommendations. These methods enable the system to catch the subtle preferences of users and suggest music that better matches their unique interests and moods.

The methodology includes preprocessing the datasets to make a single aggregrated dataset, using K-means clustering and Principal Component Analysis (PCA) to improve the representation of features, and using user interaction matrices to make collaborative filtering work better. The evaluation of the recommendation system is done by conducting user satisfaction surveys. The content-collaborative and LLM-based models have shown considerable performance, with an average satisfaction rating of around 4.2 out of 5.

This thesis discusses the implications of these findings for key research questions concerning the effectiveness of integrating LLMs and embeddings in music recommendation systems. The study provides a theoretical foundation for future research and prospective enhancements in music recommendation.

Keywords: music recommendation system, embeddings, llm, collaborative filtering, content-based filtering, contex-based filtering

Acknowledgements

I want to extend my deepest thanks to all those who enabled me to complete this thesis. Special thanks are due to my supervisors, **Bernt Arild Bremdal**, **Shayan Dadman**, and **Kalyan R Ayyalasomayajula**, whose invaluable guidance and resources were instrumental in my research.

Additionally, I am deeply grateful to my parents and siblings, whose support and encouragement have enabled me to pursue my academic goals. Their faith in my capabilities has been a constant strength throughout my educational journey.

Table of Contents

List of Figures

List of Tables

1 Introduction

Music is one of the most essential elements for a human being as it offers emotional expression and cultural connection. Open access to music platforms has led to the rise of music recommendation systems, whose goal is to ensure that people can listen to music of their preferences. The objective of matching users with music that genuinely suits their tastes is a dynamic challenge in the digital age. Collaborative filtering (CF) and content-based models (CBM) have addressed this challenge of music recommendation systems effectively [1]. Collaborative filtering suggests music by analyzing users' listening habits with similar tastes. Content-based filtering recommends tracks based on the characteristics of the music itself, like audio features, genres, and popularity. On a large dataset of songs, the collaborative recommendation system faces data sparsity from where the cold start problem starts[2]. Collaborative filtering also has a bias that tends to recommend popular songs, creating long tail problems[3]. These issues generate a crucial gap in current recommendation systems: their struggle to recommend new or less noticeable tracks within the vast landscape of available music. They typically need help interpreting and understanding the high-level semantic relationships between user behavior and profile[4]. Recent studies show that context and content information is crucial for personalized recommendations and addressing data sparsity problems [5]. By integrating LLMs, recommendation systems can provide personalized and relevant recommendations by understanding semantics better [6]. Also, categorizing music is a significant challenge since users have diverse musical tastes depending on mood and other factors like nationality, ethnicity, etc. These situations can leave users and artists feeling disconnected, especially new ones. It is crucial to address these limitations to change the musical dynamics of how listeners listen to music while encouraging new artists.

Hence, this thesis aims to bridge these gaps by developing a hybrid music recommendation system that leverages collaborative and content-based models enhanced by the sophisticated capabilities of large language models. This thesis also aims to refine the accuracy and personalization of music recommendations.

Spotify[7] is one of the leading music streaming platforms, having more than 100 million songs and is available in 184 countries. Spotify also offers extensive research resources through its API system[8]. The Spotify Web API provides access to rich audio features, which can help build a recommendation system. Leveraging these musical features, one can develop hybrid models that use both content and collaborative filtering. Also, these features can be further used in query processing for LLM-based filtering, which enhances mood-specific recommendations. In this thesis, two primary databases[9][10] have been used to develop two unique hybrid music recommendation systems that understand the user's semantic musical essence. One method provides song recommendations by parsing user queries and leveraging the LLM's semantic analyzer. While the other method covers content-based filtering and collaborative filtering for music recommendation,, incoporating embeddings. These databases contain musical features that are essential to completing the study. The study also focuses on:

1. Review existing methodologies and works relevant to musical recommendation systems.

2. Understanding Spotify Musical Features and their underlying meaning.

3. Preparing databases to ensure they contain relevant and comprehensive musical features necessary for effective implementation of recommendation algorithms.

4. Building a unique hybrid recommendation method incorporating content and collaborative filtering for songs and playlist recommendations, utilizing user interaction matrix and dimensionality reduction techniques such as Singular Value Decomposition (SVD) and Cluster information.

5. Utilizing LLMs to Construct a user query-based recommendation approach that captures the musical semantic essence of the users along with traditional content-based filtering.

6. Evaluate the recommendation system performance based on user satis-

faction.

For a detailed description of the tasks used in this study, please follow the task description provided in Appendix .1, available as a separate PDF file. The structure of this thesis is designed to address the critical aspects of music recommendation systems. Chapter 2 *Background and Literature Review* reviews existing music recommendation systems, highlighting their advantages and disadvantages and positioning the research within the music recommendation field. Chapter 3, *Methodology*, provides a detailed exposition of the methodological approach into two sections. *Section 3.1, Data Processing Methodology:* which begins with data preparation and management, followed by an in-depth exploration of feature analysis and dimensionality reduction techniques such as PCA. This section also describes the use of clustering methods like K-means and the strategic merging of genre data and user interaction data to enhance content-collaborative-based recommendations. *Section 3.2, Development of Recommendation Systems:* describes user interaction analysis and developing different recommendation algorithms, like content-collaborative recommendation systems, large language models and using them in query-based recommendation systems, and collaborative playlist generation methods. Chapter 4 *Results:* is split into detailed analyses of the processed data and the operational recommendation systems. *Section 4.1, Data Processing Results:* Presents the results from the initial data processing phase, focusing on the outcomes of feature analysis and the efficiency of clustering implementations. *Section 4.2, Recommendation Model Results:* A survey assessment was conducted to check the effectiveness of the playlist recommendation system using collaborative filtering, the hybrid song recommendation system combining content and collaborative approaches, and the advanced query-based system utilizing LLMs. Chapter 5 Discussion and Future Work: discusses the research questions, objectives, obstacles, and future improvement areas. Chapter 6 Conclusion: Concludes the thesis by summarizing the research contributions, discussing the challenges encountered, and suggesting future research avenues that could further the field of music recommendation.

2 Literature Review

This chapter provides an organized overview of the literature on music recommendation systems, specifically focusing on developing and integrating different techniques. This section starts by overviewing various hybrid collaborative filtering approaches. The following section is the Hybrid Algorithm for Content and Collaborative Filtering Techniques Utilizing SVD. The next section focuses on context-aware music recommendation systems, which employ user-specific situations to enhance and customize music selections. This conversation emphasizes the need to use contextual data to enhance the relevancy and customer happiness of suggestions greatly. The next section examines the function of sentiment analysis using BERT, highlighting its efficacy in extracting profound semantic and emotional understanding from user interactions. Furthermore, the following sections evaluate the use of machine learning in grouping Spotify audio information to improve the accuracy of music classification, hence supporting the methodological approach employed in our study. Finally, the last concludes by examining how Large Language Models (LLMs) contribute to advancing recommendation systems by increasing their complexity and sophistication. This article examines the potential of better natural language processing skills to revolutionize recommendation practices. It discusses the positive outcomes and obstacles of integrating these capabilities.

2.1 Overview of Hybrid Collaborative Filtering Systems

Hybrid collaborative filtering (CF) systems have been created to overcome the inherent constraints of conventional CF methods, including sparsity, cold start, gray sheep, and scalability problems. These systems combine collaborative filtering with other recommendation algorithms to improve their performance. This literature review analyzes several hybrid collaborative filtering (CF) systems, including their methodology, findings, and limits.

Data Collection and Selection The research examined 163 articles from many electronic journal databases, such as ACM Portal, IEEE/IEE Library, Science Direct, Springer Link, Emerald Insight, ProQuest, Wiley online Library, and JSTOR. The publications, ranging from 1994 to 2014, were selected based on criteria such as the significance of the topic, contributions made, number of citations, publishing impact factors, and recentness.

Categorization of Hybrid Collaborative Filtering Systems The hybrid collaborative filtering (CF) systems were categorized into several groups according on the recommendation methodologies they included with CF. The following categories are included:

The hybrid CF systems were classified into different categories based on the recommendation approaches they integrated with CF. These categories include: *(1) Integration of Collaborative Filtering (CF) with Content-Based (CB) Filtering, (2) Integration of CF with Demographic (DM) Filtering, (3) Integration of CF with Knowledge-Based (KB) Systems, (4) Integration of CF with Semantic-Based (SB) Systems, (5) Integration of CF with Context-Aware (CA) Systems, and (6) Integration of Different CF Algorithms*.

1. **Integrating Collaborative Filtering with Content-Based Filtering:** Hybrid systems that integrate collaborative filtering (CF) with content-based (CB) methods have the objective of addressing the issues of sparsity, cold start, and gray sheep concerns in CF. These systems use the advantages of content-based filtering by using item characteristics and user preferences. Some examples of recommender systems include Claypool et al.'s weighted hybrid recommender for online newspapers and the PTV system for TV program suggestions[11, 12] Nevertheless, these systems have obstacles, such as the issue of new users and the constrained variety of suggestions.

2. **Integration of collaborative filtering (CF) with demographic filtering**

(DM): It effectively solves the issue of new user problem by utilizing demographic data to provide personalized suggestions. The systems suggested by Song et al. utilize a combination of demographic similarity and user-based collaborative filtering to improve the accuracy of recommendations[13]. However, acquiring demographic data can be challenging, and these systems continue to face difficulties in dealing with outliers and new item issues.

3. **Integrating Collaborative Filtering with Knowledge Base Systems:** Hybrid systems that combine CF with KB systems can handle sparsity and cold start problems by leveraging a knowledge base to provide recommendations. Examples include Towle and Quinn's weighted hybrid approach and the EntreeC system, which uses KB techniques to bootstrap the CF engine [14, 15]. These systems, however, require extensive knowledge engineering.

4. **Integrating Collaborative Filtering with Semantic-Based Systems:** Semantic-based hybrid systems improve the quality of recommendations by taking into account the semantic connections between products and users. The systems that Ceylan and Birturk created make suggestions more accurate in datasets with little information and effectively deal with the problems of cold start and gray sheep [16]. Nevertheless, they also necessitate significant knowledge engineering endeavors.

5. **Integrating Collaborative Filtering with Context-aware Filtering Systems:** Context-aware hybrid systems utilize contextual information to enhance the relevancy of recommendations. Systems such as the Smart Radio music playlist recommender adjust its suggestions according to user circumstances, such as location and time [17]. Although they perform well in dynamic settings, they do not completely address issues related to sparsity, cold start, and gray sheep concerns.

6. **Integration of Different CF Algorithms:** The combination of memory-based and model-based collaborative filtering (CF) approaches exploits the scalability of model-based methods while maintaining the accuracy of memory-based methods. Xue et al. have suggested systems that utilize clustering of user data to improve the quality and scalability of recommendations [18]. Nevertheless, the implementation of these systems is intricate and costly.

Limitations and discussion Hybrid CF systems, despite their progress, encounter certain constraints:

1. **Requirements for Knowledge Engineering**: Systems that combine knowledge-based (KB) and semantic-based (SB) techniques require a substantial amount of knowledge engineering, which can be demanding in terms of resources.

2. **Data Privacy Concerns**: The process of gathering demographic and contextual data can give rise to concerns over the protection of personal information and can be difficult to accomplish.

3. **Scalability Issues**: Although model-based techniques enhance scalability, they frequently do so at the cost of accuracy.

4. **Complexity and Implementation Costs**: Hybrid systems possess intrinsic complexity and can incur significant expenses during development and maintenance.

Alignment with Current Research The study by Gohari et al. goes into great detail about collaborative hybrid systems. It classifies these systems, talks about their pros and cons, and compares how well they deal with the main problems of CF[19]. Gohari et al. talk about the benefits of combining semantic-based methods with collaborative filtering to get better and more varied suggestions in changing settings. Our research extends to selecting the hybrid models appropriate for building our music recommendation system.

2.2 Hybrid Algorithm Utilizing Content and Collaborative Filtering

The study "Hybrid Algorithm Based on Content and Collaborative Filtering in Recommendation System Optimization and Simulation" [20] explores the integration of content-based filtering and collaborative filtering techniques to enhance recommendation systems. The purpose of this hybrid strategy is to tackle prevalent challenges such as the scarcity of data, difficulties in starting from scratch, and the requirement for suggestions in real time.The purpose of this hybrid strategy is to tackle prevalent challenges such as the scarcity of data, difficulties in starting from scratch, and the requirement for suggestions in real time.

Methods The hybrid recommendation algorithm suggested in this study incorporates the advantageous aspects of content-based and collaborative filtering techniques.

1. **Content-Based Filtering**:

    - This approach to filtering involves analyzing the content of items to provide recommendations based on their similarity to items that the user has already shown interest in. object User interests are derived from their surfing history and shopping data. The analysis of the relationship between user preferences and item characteristics using algorithms like TF-IDF and cosine similarity determines characteristics.

    - A content-based recommendation module processes the data to create a profile of user interests.

2. **Collaborative Filtering**:

    - Collaborative Filtering is a method used to make predictions or recommendations by collecting and analyzing data from several users or sources. The user rating data, in conjunction with the current access sequences, is utilized to ascertain users and objects that exhibit comparable characteristics. The collaborative filtering module utilizes the commonalities between users to forecast ratings and produce suggestions.

    - The Singular Value Decomposition (SVD) is used in the collaborative filtering module to decrease the dimensionality of the user-item interaction matrix, hence improving computing efficiency and accuracy.

3. **Synthesis of Both Approaches**: The final recommendation list is produced by aggregating the results from both modules using a weighted sum computation. This comprehensive approach alleviates the constraints of each separate method and enhances the overall quality of recommendations.

Results The testing resuls demonstrated several significant advantages of the hybrid approach:

1. **Enhanced Precision**: The hybrid approach demonstrated a substantial decrease in Mean Absolute Error (MAE) in comparison to solo collaborative filtering techniques. As the quantity of training data rose, the MAE values declined, suggesting improved prediction accuracy.

2. **Addressing Data Sparsity and Cold Start**: The technique efficiently resolves the problem of data sparsity by integrating content-based filtering with collaborative filtering. Unrated goods can still be recommended based on their content features, which helps address the cold start problem.

3. **Performance using Singular Value Decomposition (SVD) and Clustering**: Using Singular Value Decomposition (SVD) to reduce the number of dimensions in the collaborative filtering module speeds up computations and improves the accuracy of recommendations. In addition, K-means clustering is utilized to categorize users according to their attributes, resulting in improved computational efficiency and increased precision in suggestions.

4. **Improved Recommendation Quality**: The hybrid technique exhibited superior recommendation quality across many datasets, including the MovieLens dataset. It achieved a compromise between the requirement for processing in real-time and the goal of maintaining a high level of suggestion accuracy.

Limitations Although the research has notable merits, it also admits some limitations:

1. **Computational Complexity**: The incorporation of two filtering methodologies and the use of clustering contribute to the computational intricacy. Although the approach is efficient, it may nevertheless necessitate substantial resources for really big datasets.

2. **Parameter Sensitivity**: The sensitivity of the parameters is being assessed. The efficacy of the hybrid algorithm is contingent upon the meticulous calibration of diverse parameters, including the weights assigned to amalgamate recommendations and the quantity of clusters in K-means.

3. **Offline Evaluation**: The study predominantly depends on offline datasets for assessment. Conducting real-world trials and validating the suggested strategy through online experiments are essential to completely proving the practical usefulness of the method.

Alignment with Current Research The hybrid recommendation algorithm, which combines content-based and collaborative filtering techniques, offers a strong solution to overcome the constraints of conventional recommendation systems. The algorithm enhances accuracy, addresses data sparsity, and offers high-quality suggestions by efficiently merging the advantages of both techniques. This method is especially advantageous for real-time applications where both performance and accuracy are crucial. Singular Value Decomposition (SVD) to lower dimensionality in collaborative filtering can be looked forward to when creating a hybrid recommendation system.

2.3 Overview of Context-Aware Music Recommendation Systems

Context-aware recommendation systems have become increasingly important in personalized music recommendation, aiming to provide users with music that matches their contextual preferences. This literature review focuses on a study by Wang et al. that proposes a novel approach for context-aware music recommendation, explaining its methodology, findings, and limitations [21].

Method The study utilizes a dataset obtained from Xiami Music, which consists of 4,284,000 recordings of music playback. These records contain 361,861 unique musical compositions that 4,284 different users created. The suggested method acquires reduced-dimensional representations of music compositions based on users' past playing sequences and information. This embedding approach, which draws inspiration from neural language models, considers the listening history of each user as a "sentence" and regards each music piece as a "word" inside that sentence.

The historical playing sequence of each user is analyzed using a sliding window approach to provide training data. This analysis takes into account both the local context (music pieces in close proximity) and the global context (total listening history). The embeddings undergo refinement using a regularization procedure that takes into account metadata such as album and artist information. This ensures that musical pieces with similarities have embeddings that are near each other.

Contextual Preference Inference and Recommendation The approach encompasses both overarching and situational user preferences. The global preference is determined by analyzing the user's whole listening history, whereas the contextual preference is influenced by the user's most recent music choices. The likelihood of a user having a preference for a certain music piece is calculated by measuring the cosine similarity between the music's embedding and the user's overall and situational preferences. Subsequently, music compositions are assessed based on this likelihood and suggested to the user.

Experimental Design and Evaluation The effectiveness of the suggested method was assessed using an actual dataset and four metrics: accuracy, recall, F1 score, and hitrate. The dataset was partitioned into training and test sets using 5-fold cross-validation. The method's efficacy was evaluated by comparing it to three baseline approaches: Injected Preference Fusion (IPF) based on Temporal Recommendation, Bayesian Personalized Ranking (BPR), and FISMauc (FISM). [22, 23, 24]

Results and Discussion The experimental results demonstrated that the proposed approach outperformed the baseline methods significantly. For instance, the proposed method showed a relative improvement in F1 score of around 96.4%, 69.7%, and 42.6% over BPR, FISM, and IPF, respectively. These improvements highlight the effectiveness of incorporating contextual preferences into music recommendations. Additionally, the study found that embeddings learned by the proposed model effectively captured the similarities between music pieces, ensuring relevant recommendations.

Despite its promising results, the study has several limitations:

1. **Complexity and Training Time**: The sliding window technique and embedding learning process are computationally intensive, requiring significant training time, especially with larger window sizes.

2. **Parameter Tuning**: The performance of the model heavily depends on the appropriate setting of parameters, such as the combination weight (β) and the embedding dimension. Inadequate calibration might result in less than ideal outcomes.

3. **Offline Evaluation**: The study predominantly relied on metrics for assessment conducted without direct user involvement, which may not comprehensively measure user satisfaction. Potential future research might entail conducting online experiments to gain a deeper understanding of the immediate influence of recommendations on the user experience.

The study by Wang et al. presents an effective approach for context-aware music recommendation by leveraging embeddings learned from historical playing sequences and metadata. The method successfully integrates users' general and contextual preferences, resulting in superior recommendation performance compared to traditional methods. Future research could focus on enhancing the computational efficiency of the model and validating its effectiveness through online user studies.

Alignment with Current Research Wang et al.'s system employs low-dimensional embeddings for music pieces, similar to word embeddings in natural language processing, derived from users' historical interactions and metadata. These embeddings are utilized to model both broad and specific contextual preferences of users, which is particularly relevant to our research focus. Our work extends this approach by incorporating advanced clustering techniques with these embeddings to refine the recommendation accuracy based on user-specific contexts, enhancing the system's adaptability to real-time user preferences and activities.

2.4 BERT-Based Sentiment Analysis

An essential aspect of software engineering is sentiment analysis, which plays a critical role in comprehending the feelings and views of developers. This analysis is vital in enhancing tools and project management. The research article titled "BERT-Based Sentiment Analysis: A Software Engineering Perspective," authored by Batra et al. investigates the application of transformer models in analyzing sentiment in software engineering texts[25].

Data Collection The study employed datasets from GitHub, Jira, and Stack Overflow, comprising comments and postings pertaining to software engineering. The datasets underwent augmentation through the use of lexical substitution and back translation techniques in order to enhance variety and optimize model training.

DistilBERT Model DistilBERT is a condensed iteration of BERT that maintains 97% of BERT's language comprehension capabilities, while being 40% smaller and 60% quicker. The model is trained to emulate BERT by acquiring knowledge from the probability generated by BERT before to the final activation function. This technique guarantees that DistilBERT maintains the fundamental characteristics of BERT while minimizing computing burden.

- Fine-Tuning: The DistilBERT model underwent fine-tuning using datasets related to software engineering. The process of fine-tuning entails the addition of an untrained layer to the pre-trained DistilBERT model, followed by training it for sentiment classification across a limited number of epochs.

- **Training and Testing**: The datasets were divided into separate sets for training, validation, and testing purposes. During the training phase, early pausing was implemented to minimize overfitting. The models were then assessed using precision, recall, and F1-score measures.

Additional BERT-Based Models

In addition, the study optimized several BERT variants (BERT, RoBERTa, and ALBERT) using the same datasets. In addition, a collective technique was used to integrate the predictions of various models to improve performance. The ensemble employed a weighted voting mechanism that relied on the confidence ratings derived from the final softmax layer of each model.

Results The experimental results demonstrated that DistilBERT performs exceptionally well for sentiment analysis in software engineering:

Table 2.1: Performance Comparison of Models[25]

Model	GitHub (F1-score)	Stack Overflow (F1-score)	Jira (F1-score)
DistilBERT	0.92	0.86	0.91
RoBERTa (Fine-Tuning)	0.91	0.85	0.83
Ensemble Model	0.92	0.92	0.90

DistilBERT Performance

- **Effectiveness**: DistilBERT achieved high F1-scores across all datasets, indicating its effectiveness. For instance, it recorded F1-scores of 0.92 on GitHub, 0.86 on Stack Overflow, and 0.91 on Jira. These results represent a significant improvement over existing tools.

- **Efficiency**: The reduced size and faster inference time of DistilBERT make it a practical choice for deployment in resource-constrained environments. It strikes a balance between performance and computational efficiency, making it suitable for real-time applications.

- **Improvement Over Baselines**: DistilBERT outperformed traditional sentiment analysis tools like SentiStrength, SentiCR, and Senti4SD, showing improvements of 6–12% in F1-scores. This highlights the advantage of using transformer-based models in this domain.

Performance of Other Models

- **Fine-Tuning Results**: Fine-tuning BERT variants yielded substantial improvements in F1-scores across all datasets. For instance, the RoBERTa model achieved F1-scores of 0.91 on GitHub, 0.85 on Stack Overflow, and 0.83 on Jira.

- **Ensemble Model Performance**: The ensemble approach further enhanced performance, achieving an F1-score of 0.92 on GitHub, 0.92 on Stack Overflow, and 0.90 on Jira. The weighted voting scheme effectively combined the strengths of individual models, resulting in more accurate predictions.

Alignment with Current Research The study by Batra et al. demonstrates the effectiveness of DistilBERT as a potent tool for sentiment analysis in the field of software engineering. It effectively combines superior performance with computational economy. Through the process of fine-tuning this compressed model using datasets relevant to a particular area, the authors were able to make noteworthy enhancements compared to previous techniques. The results indicate that DistilBERT is a feasible choice for sentiment analysis applications that require real-time processing, especially in situations with limited resources. Leveraging this portability and computationally friendly pattern of DistilBERT, we can look forward to utilize it to analyse sentiment of user for our query-based song recommendation system.

2.5 Literature Review: Machine Learning-Based Clustering Utilizing Spotify Audio Features

The research paper titled "Machine Learning Based Clustering Using Spotify Audio Features"[26] explores the utilization of several clustering algorithms to categorize music effectively based on Spotify audio attributes. The main goal is to examine and categorize songs into distinct clusters according to their auditory characteristics, including danceability, energy, loudness, valence, and pace. This study employs the K-means, Fuzzy C-means (FCM), and Possibilistic C-means (PCM) algorithms and assesses their performance by utilizing the silhouette score as the assessment metric.

Methods The study utilizes three clustering techniques to classify the Spotify audio characteristics dataset:

1. **K-means Clustering**: K-means clustering is a method used to partition a dataset into K distinct clusters, where each data point belongs to the cluster with the nearest mean. The K-means algorithm is a commonly employed clustering method that divides data into K separate groupings. The objective is to reduce the variation inside each cluster and increase the variation across clusters. The technique progressively modifies cluster centroids to get optimum differentiation.

2. **Fuzzy C-means (FCM)**: object FCM, or Fuzzy C-means, is a modified version of the K-means algorithm that permits data points to be assigned to several clusters with different membership levels. This technique employs fuzzy logic principles to allocate membership values, making it well-suited for datasets characterized by overlapping clusters. The list ends.

3. **Possibilistic C-means (PCM)**: PCM overcomes the constraints of FCM by loosening the need for the total of row values, which enables greater freedom in assigning cluster memberships. This approach is very advantageous for managing noise and outliers in the dataset.

Results The study's results are shown in Table 1, which displays the silhouette scores for each clustering procedure for various values of n (number of clusters).

Table 2.2: Comparison of Silhouette Scores for Clustering Algorithms[26]

Clustering Algorithm	Silhouette Score
K-means (n=6)	0.465
K-means (n=5)	0.426
K-means (n=4)	0.425
FCM (n=6)	0.137
FCM (n=5)	0.20752
FCM (n=4)	0.235
PCM (n=6)	0.35177
PCM (n=5)	0.35177
PCM (n=4)	0.35177

1. The **K-means** algorithm obtained a silhouette score of 0.465 with $n = 6$, suggesting distinct clusters with low overlap.

2. The FCM algorithm exhibited a decrease in performance as the value of n grew, reaching its greatest silhouette score of 0.235 when n was equal to 4.

3. The silhouette score of 0.35177 for **PCM** remained constant for all values of n, demonstrating consistent performance.

Limitations

1. **Computational Complexity**: The utilization of various clustering techniques and the handling of extensive datasets necessitate substantial computer resources.

2. **Parameter Sensitivity**: The effectiveness of clustering algorithms is contingent upon the selection of parameters, such as the quantity of clusters (n), which might impact the caliber of the outcomes. The itemized list ends.

3. **Evaluation Metrics**: The silhouette score in isolation does not offer a comprehensive assessment of clustering performance. Additional measures and specialized knowledge in certain domains are required to understand the results accurately. The list is complete.

Alignment with Current Research The study shows that K-means works better than FCM and PCM in grouping Spotify audio characteristics, as indicated by higher silhouette scores. The results indicate that K-means is a reliable and effective technique for classifying music, especially when the appropriate number of clusters is used. Nevertheless, selecting a clustering technique should consider the particular objectives and attributes of the dataset. Clustering patterns can improve music recommendation systems by offering more individualized and pertinent choices. K-means can play vital role while clustering musical from spotify and can be further used on content based recsys.

2.6 Recommender Systems in the Era of LLMs

The research paper titled "Recommender Systems in the Era of Large Language Models (LLMs)" [27] investigates the profound influence of LLMs on recommender systems (RecSys). Traditional deep neural network-based recommendation methods have limitations in understanding user interests and capturing textual information. However, LLMs like GPT-4 and ChatGPT offer new opportunities to enhance RecSys by leveraging their advanced language understanding and generation capabilities. This research examines the utilization of Language Models (LLMs) to enhance the effectiveness and adaptability of recommender systems.

Methods The research classifies the incorporation of LLMs into RecSys into three main parts:

1. **Pre-training**: Prior training: object LLMs undergo pre-training using extensive and varied textual datasets to acquire knowledge of broad language patterns and semantics. This phase include activities such as Masked Language Modeling (MLM) and Next Token Prediction (NTP), which aid in the comprehension of context by the models and the production of coherent text. The item list ends.

2. **Fine-tuning**: Adjusting or optimizing a system or process to get optimal performance or results. Following pre-training, Language Models (LLMs) undergo fine-tuning using task-specific datasets that consist of user-item interaction data and textual side information. Fine-tuning can be categorized as either full-model or parameter-efficient (PEFT), depending on whether all model parameters or simply a selection of parameters (such as adapters) are changed. The itemized list ends.

3. **Prompting**: Item prompting utilizes task-specific templates to customize LLMs for downstream recommendation tasks without modifying the model parameters. In-context learning (ICL) and chain-of-thought (CoT) prompting are ways to teach language models (LLMs) how to do tasks by using the context and logical steps given in the prompts.

Results The paper emphasizes several improvements in the utilization of LLMs for recommender systems.

1. **Improved Text Comprehension**: LLMs greatly enhance the capacity to comprehend and produce text, rendering them very efficient for jobs that include extensive textual data such as reviews, item descriptions, and user profiles. This improves the quality of recommendations by offering choices that are more appropriate in the given environment. The item list ends.

2. **Generalization and Flexibility**: LLMs exhibit remarkable generalization skills, enabling them to adjust to different recommendation tasks with minimum fine-tuning. Their aptitude to excel in zero-shot or few-shot scenarios diminishes the necessity for extensive, task-specific datasets. The itemized list ends.

3. **Methods of Integration**: The study explores novel methodologies such as mutual regularization and soft+hard prompting, which integrate collaborative filtering with content-based approaches to improve the precision and resilience of suggestions. The list ends.

4. **Enhancements in Performance**: There is evidence that RecSys models with LLM do better than standard models at a number of tasks, such as making top-K suggestions, rating predictions, and conversational suggestions. Models such as P5 and TALLRec have shown substantial enhancements in recommendation precision and user contentment.

Limitations and Discussion Although there have been significant gains, the report notes the presence of several difficulties.

1. **Computational Resources**: Training and optimizing Language Models (LLMs) need significant computing resources, which might restrict their availability and scalability in practical scenarios.

2. **Data Privacy and Ethics**: The utilization of extensive user data for the purpose of training Language Models (LLMs) gives rise to apprehensions over data privacy and ethical ramifications. It is essential to guarantee the responsible and safe use of data.

3. **Hallucination and Reliability**: LLMs have the ability to occasionally provide information that seems believable but is really erroneous, which is referred to as hallucination. This problem presents potential dangers in situations where important recommendations are made, requiring strong procedures to confirm and authenticate the results of the model.

The incorporation of extensive language models into recommender systems signifies a notable progression in the domain, providing superior text comprehension, greater ability to apply knowledge to different situations, and increased adaptability to diverse jobs. Despite persistent obstacles such as limited computational resources, data privacy concerns, and model dependability issues, continuous research and inventive methods are consistently expanding the capabilities of LLMs in improving recommendation systems.

3 Methodology

Music recommendation systems play a vital role in helping users navigate the extensive collection of accessible music. However, these systems frequently encounter major challenges that can impact the user's enjoyment and overall performance. The "long tail" problem and the "cold start" problem are two significant challenges frequently faced. The long tail problem is when many niche songs receive minimal or no attention, resulting in many potentially good music remaining unknown by listeners. Conversely, the cold start problem occurs when the algorithm has insufficient data on newly introduced music or users, resulting in inaccurate suggestions. In addition, the system must do intricate semantic evaluations to provide individualized music recommendations that relate to each listener's distinct musical preferences and emotions.

To tackle these challenges, we propose the development of two separate recommendation systems:

1. **Hybrid Song Recommendation System:** This system combines Content-Based and Collaborative Filtering techniques to solve the Cold Start and Long tail problems.

   (a) Content-Based Filtering: Aligns user input with genres and artists.

   (b) Collaborative Filtering: Evaluate the similarities of songs and user participation patterns and then customize suggestions.

   (c) Input: Song name chosen by the user.

2. **Query-Based Song Recommendation System:** This method combines Content-Based and Context-Based Filtering, enhanced by the LLM base model DistilBERT for more profound semantic analysis. This method addresses the cold start and thoroughly analyzes a user's semantic nature.

   (a) Content-Based Filtering: Emphasizes the matching of genre and mood based on the user's query.

   (b) Context-Based Filtering: Adjusts suggestions by considering the user's historical preference, like decades, and the emotional tone of their input, which is assessed using LLM.

   (c) Input: User prompt indicating distinct musical tastes or moods.

Both recommendation systems utilize specific databases and advanced modeling approaches to improve their functionality:

1. **Primary Database:** The Spotify Musical (Data.csv)[10] open source database includes extensive musical feature values, such as acoustics, valence, and other relevant metadata, along with its song name and artist name. These values are the cornerstone of enabling content-based filtering procedures in both systems.

2. **Secondary Database:** Spotify Playlist Dataset (Spotify_UI_DB)[9] is another open source database derived from users sharing their "now playing" tracks on social media; this dataset provides real-time interaction data, including information on users, their playlists, and the tracks within. It is crucial for the collaborative filtering aspect of the Hybrid Recommendation System.

3. **LLM Model:** The DistilBERT-based LLM is a model from Hugging Face's transformers library. It is a more compact version of the BERT model and has been specially optimized for sentiment analysis. It greatly improves the Query-Based Recommendation System by studying and understanding user inquiries' emotional and contextual subtleties. This feature enables the system to customize recommendations that align more profoundly with the user's mood and preferences.

These components jointly improve the recommendation systems by ensuring that recommendations are relevant and emotionally suited to the user's condition. The figure below illustrates the integration of content-based and collaborative filtering techniques in our Hybrid Song Recommendation System and semantic analysis using the DistilBERT-based LLM in our Query-Based Song Recommendation System.

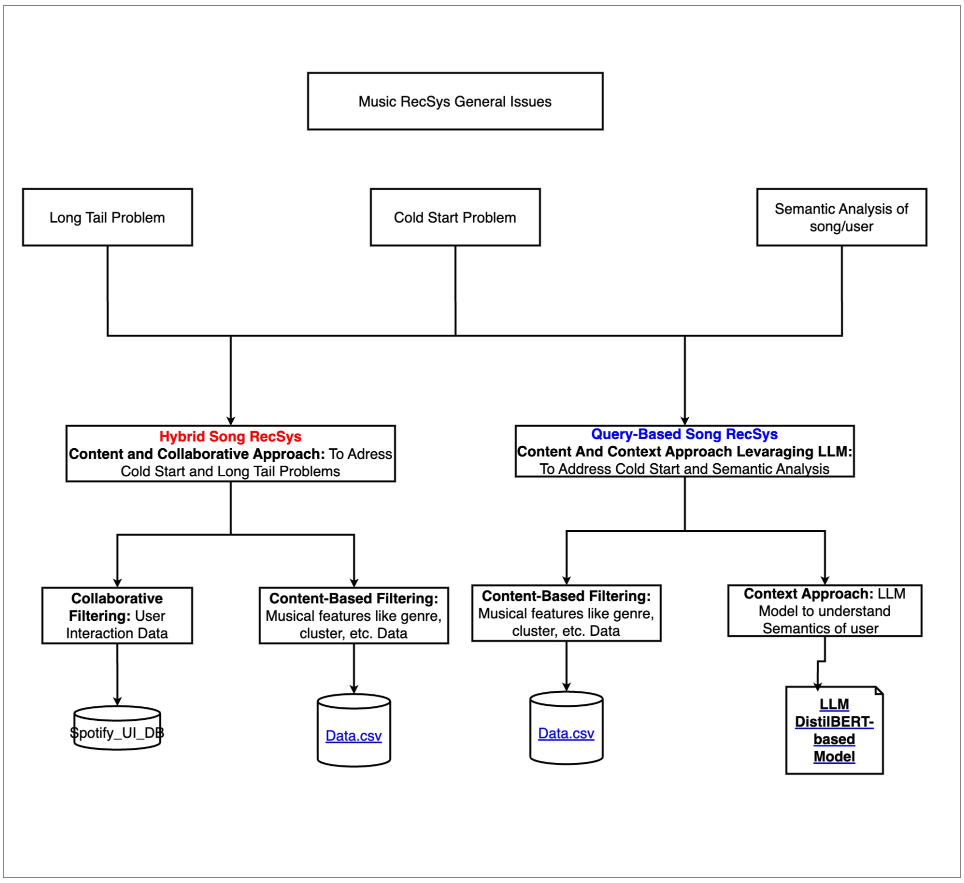

Music RecSys General Issues
Long Tail Problem
Cold Start Problem
Semantic Analysis of song/user
Hybrid Song RecSys
Content and Collaborative Approach: To Adress Cold Start and Long Tail Problems
Query-Based Song RecSys
Content And Context Approach Levaraging LLM: To Address Cold Start and Semantic Analysis
Collaborative Filtering: User Interaction Data
Content-Based Filtering: Musical features like genre, cluster, etc. Data
Content-Based Filtering: Musical features like genre, cluster, etc. Data
Context Approach: LLM Model to understand Semantics of user
Spotify_UI_DB
Data.csv
Data.csv
LLM DistilBERT-based Model

Figure 3.1: Schematic representation of the proposed music recommendation systems, illustrating how different data sources and the DistilBERT-based LLM are utilized to tackle challenges such as the long tail and cold start problems and to understand user semantic needs.

The subsequent sections will delve deeper into the data processing techniques utilized (**Section 3.1**) and elaborate on the specific methodologies employed in the recommendation systems (**Section 3.2**), followed by a detailed evaluation of the systems' performances (**Section 4**). All the source code for this thesis is provided here: [28]

3.1 Data Processing Methodology

This section describes the development and integration of specialized datasets named `data.csv`[10] and `Spotify_UI.csv`[9].

`Aggregrated.csv` was designed by incorporating these two databases to improve the performance of our hybrid recommendation systems. We carefully preprocess and arrange these datasets to meet the operational requirements of our recommendation methodology: content-based and collaborative filtering techniques and Query-Based LLM filtering. These databases are critical in deploying our advanced, hybrid recommendation algorithms by offering a solid data basis. The preprocessing method is designed to optimize the databases to efficiently support the system in giving personalized music recommendations.

1. Data Preparation and Management

   (a) **Data Preparation**: The primary dataset, named `data.csv`[10], was collected from Kaggle, an open-source platform for data science projects and competitions. The dataset was created and made publicly available by user Vatsal Mavani. This database includes comprehensive data on Spotify tracks, featuring a variety of musical attributes such as genre, popularity, year, acousticsness, energy, and other relevant metrics essential to us for content-based and Context-based filtering processes. The datasets were accessed by visiting to the kaggle. All the terms and conditions were read before downloading the dataset.

   (b) **Data Preprocessing:** Theis Database has undergone thorough preprocessing, which involves normalization, handling missing values, and feature selection. This phase is responsible for preprocessing the data for precise and efficient analysis.

2. Feature Analysis and Dimensionality Reduction

   (a) **Feature Analysis and Dimensionality Reduction:** After preprocessing the data, further analysis was done using exploratory data analysis.

   (b) **Correlation Analysis of Musical Features:** To determine the connections between different musical features.

   (c) **Principal Component Analysis:** PCA was introduced to reduce the dimensionality of the features, ensuring maximum variance.

3. K - Means Clustering

   (a) **Optimal Cluster Determination:** Silhouette score was calculated to determine the optimal cluster number for the PCA-reduced Database. Furthermore, the Calinski-Harabasz Index (CHI) and Davies-Bouldin Index(DBI) are calculated to validate the well-separated clusters.

   (b) **Cluster Labels:** We use Z-score normalization on the cluster feature to enhance the understanding of every music cluster. By determining the relative relevance of each attribute inside the clusters, this analysis method makes it easier to provide more detailed and insightful descriptions for each cluster.

4. Merge Genre Data

   (a) **Additional Data Merge:** The descriptive labeled data is saved and merged with an additional genre dataset (`data_w_genre`)[10], consists of genres which are crucial to content-based filtering. This database has also been preprocessed for missing values and normalization.

   (b) **Storing Cluster Described Dataset:** The Dataset has been saved for further operations.

5. Aggregated Data with User Interaction Data

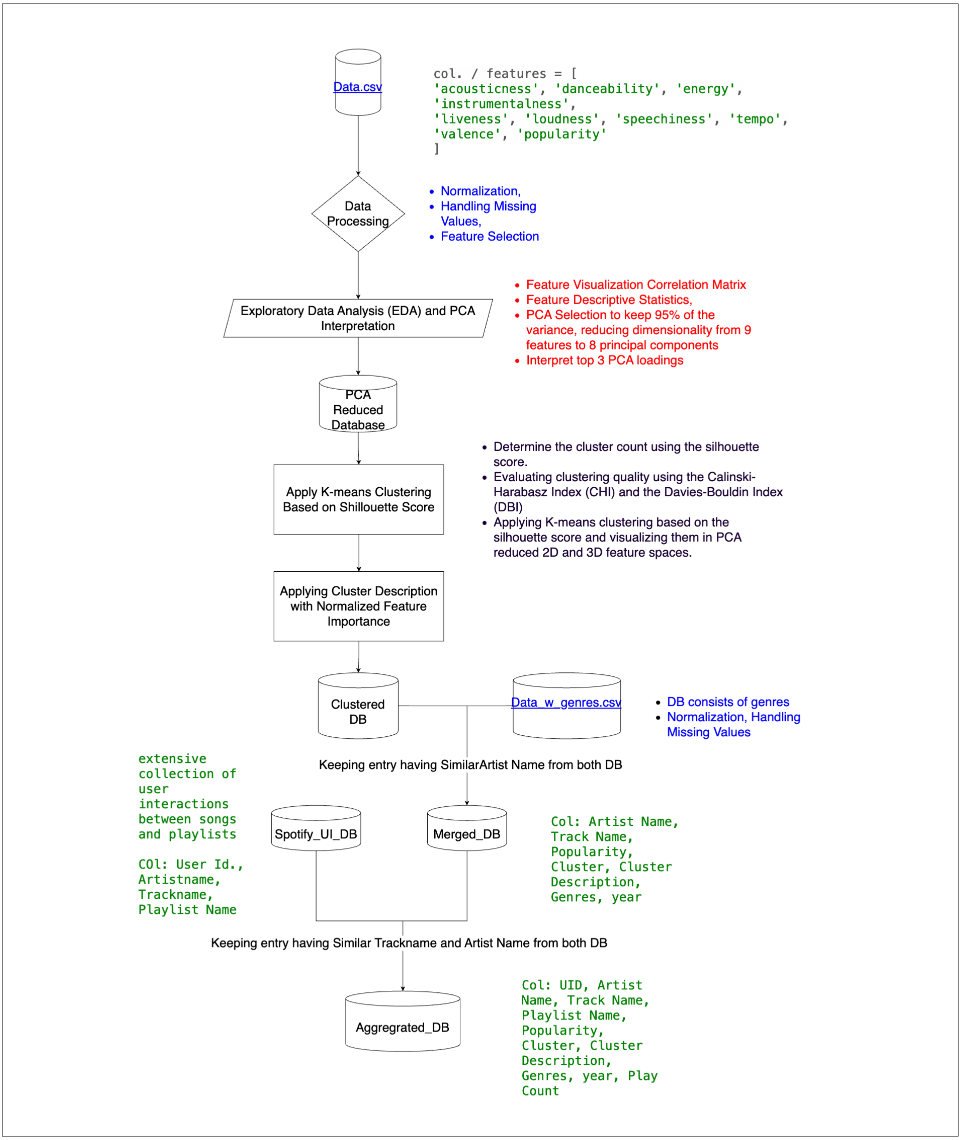
   (a) This merged dataset is further combined with the Spotify user interaction database (`spotify_playlist.csv`)[9] collected from Kaggle. This dataset captures real-time user interactions with music tracks, including data on users, their playlists, and the tracks within these playlists, which includes preprocessing steps similar to the initial datasets. These entries are essential to building a user interaction matrix and creating collaborative Filtering.

   (b) This Database was then aggregated with the merged dataset consisting of genre clusters and other important metrics for content-based filtering to create an advanced dataset named `Aggregrated.csv`.

Each component of this `Aggregrated.csv` plays a pivotal role in ensuring the music recommendation system's effectiveness and personalization, addressing the industry's core challenges.

These steps collectively form a robust approach to developing an advanced music recommendation system, as detailed in the flowchart provided in Figure

3.2.



```
col. / features = [
'acousticness', 'danceability', 'energy',
'instrumentalness',
'liveness', 'loudness', 'speechiness', 'tempo',
'valence', 'popularity'
]
```

Data.csv
Data Processing
Normalization,
Handling Missing Values,
Feature Selection
Exploratory Data Analysis (EDA) and PCA Interpretation
Feature Visualization Correlation Matrix
Feature Descriptive Statistics,
PCA Selection to keep 95% of the variance, reducing dimensionality from 9 features to 8 principal components
Interpret top 3 PCA loadings
PCA Reduced Database
Apply K-means Clustering Based on Shillouette Score
Determine the cluster count using the silhouette score.
Evaluating clustering quality using the Calinski-Harabasz Index (CHI) and the Davies-Bouldin Index (DBI)
Applying K-means clustering based on the silhouette score and visualizing them in PCA reduced 2D and 3D feature spaces.
Applying Cluster Description with Normalized Feature Importance
Clustered DB
Data_w_genres.csv
DB consists of genres
Normalization, Handling Missing Values
Keeping entry having SimilarArtist Name from both DB
extensive collection of user interactions between songs and playlists
COl: User Id., Artistname, Trackname, Playlist Name
Spotify_UI_DB
Merged_DB
Col: Artist Name, Track Name, Popularity, Cluster, Cluster Description, Genres, year
Keeping entry having Similar Trackname and Artist Name from both DB
Aggregrated_DB
Col: UID, Artist Name, Track Name, Playlist Name, Popularity, Cluster, Cluster Description, Genres, year, Play Count

Figure 3.2: Flowchart illustrating the detailed Data Processing methodological framework adopted in this thesis

3.1.1 Data Preparation and Management

We utilized the *data.csv*[10] dataset, which includes a comprehensive array of musical features and metadata. This dataset is processed using the `pandas`[29] library in Python.

Database Overview The dataset comprises a total of **170,653** rows and **19** columns, containing a comprehensive range of data points vital for the music recommendation system. The columns relevant to our analyses are listed below:

Total Rows	170,653
Total Columns	19
Key Features	Valence, Year, Acousticness, Danceability, Energy, Instrumentalness, Liveness, Loudness, Popularity, Speechiness, Tempo

Table 3.1: Overview of the dataset used in the music recommendation system.

The dataset reveals the following structure, showcasing a subset of the features available for each track:

- Valence: A measure of musical positiveness.
- Danceability: How suitable a track is for dancing.
- Energy: A perceptual measure of intensity and activity.
- Instrumentalness: The likelihood of the track being instrumental.
- Liveness: The presence of an audience in the recording.
- Loudness: The overall loudness of a track in decibels.
- Mode: The modality (major or minor) of a track.
- Popularity: The popularity of the track.
- Speechiness: The presence of spoken words in a track.
- Acousticness: A measure of the track's acoustic properties.
- Tempo: The overall estimated tempo of a track.

This initial examination of the dataset is vital for interpreting its structure and directing the next development phase across our methodology. These features will help match tracks with user preferences based on musical properties, helping us build a content-based filtering model.

Data Management and Cleaning The dataset was loaded to evaluate its structure and to identify essential features from the feature list. Maintaining the integrity and quality of our dataset is crucial before using any machine learning techniques. This section discusses the procedures used to detect and resolve any instances of missing values, duplication, and inconsistencies in our dataset. This technique improves the precision of our music recommendation engine and guarantees our analysis's dependability.

- Missing Data: We identified and removed rows with missing values to maintain the integrity of our dataset.
- Duplicate Records: We checked for and eliminated duplicate entries to ensure the uniqueness of the data.
- Validation: A verification step was conducted after cleaning to ensure the consistency and accuracy of the dataset.

The clean Dataset contained a total of **170,653** records across **19 features**, with no missing values or duplicates detected post-cleaning.

3.1.2 Feature Analysis and Dimensionality Reduction for K-means clustering

From the Database, these features were selected: acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, and popularity. These features help us understand the depth of the database and, in the long run, will guide us to develop a content-based filtering model. Again, reducing these features into PCA-reduced space will enhance the method's efficiency, and the data will be ready for k-means clustering.

Exploratory data analysis (EDA) As part of the exploratory data analysis (EDA), we visualized the distribution of important musical features to acquire insights into their characteristics and determine the preprocessing procedures needed for the K-means clustering technique. Turning these musical features into clusters will help us build an accurate and efficient recommendation system. It will save computing power along with its scalability and adaptability. Histograms were created for the following attributes: acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, and popularity. Descriptive statistics were computed to provide a foundational understanding of the dataset's characteristics. Detailed in the supplementary material, these statistics encapsulate key musical attributes' central tendencies and dispersions.

Correlation Analysis of Musical Features We conducted a correlation study to determine the connections between these selected musical features. This correlation analysis determined the degree and direction of the linear correlations between a subset of musical features. This study helps to identify any potential overlap between characteristics that impact the efficacy of the clustering method. A coefficient close to 1.0 indicates a strong positive correlation. A coefficient close to -1.0 indicates a strong negative correlation. A coefficient around 0 indicates no linear relationship.

Dimensionality Reduction with PCA To enhance the efficiency of our music recommendation system, we utilized Principal Component Analysis (PCA) on a specific set of musical attributes: ['acousticness,' 'danceability,' 'energy,' 'instrumentalness,' 'liveness,' 'loudness,' 'speechiness,' 'tempo,' 'valence']. To reduce complexity while maintaining key data features by condensing the information in the dataset into fewer dimensions.

The PCA successfully decreased the number of dimensions by choosing eight principle components that account for 95% of the dataset's variability. The variation accounted for by these components is as follows:

The percentages of variance explained by each PCA component are as follows:

- PCA Component 1 explains 34.25%,
- PCA Component 2 explains 15.33%,
- PCA Component 3 explains 12.60%,
- .....

These values indicate that the first few components capture the most significant portions of variance within the dataset, with the first three alone accounting for over 60%

These components significantly reduce the dataset's complexity, as the primary variations in musical characteristics are efficiently condensed into fewer dimensions. This decrease is especially beneficial for K-means clustering due to many reasons. Reducing the number of dimensions results in more significant and computationally efficient distance computations. Each principal component analysis (PCA) component reflects a set of attributes that together account for a particular element of the variability in the data. Utilizing principle components instead of a broader number of independent attributes allows for a more focused analysis of the clusters created by K-means.

The principal component analysis (PCA) plays a crucial role in streamlining the dataset while maintaining its essential information, hence guaranteeing the effectiveness and efficiency of the K-means clustering algorithm.

A detailed study of PCA loadings, which provides insights into the contribution of each original feature to the principal components, is included in the appendix. This additional analysis helps further interpret the dimensions retained for clustering and their implications for the recommendation system.

3.1.3 K-Means Clustering

Optimal Cluster Count Determination The optimal number of clusters was guided by evaluating silhouette scores across a range of 2-11, complemented by the Calinski-Harabasz Index and Davies-Bouldin Index for cluster validation. This technique will ensure meaningful groupings in the songs for the user.

Silhouette Score Analysis: The **Silhouette Coefficient**[30] measures the clustering quality when ground truth labels are unknown. It combines two scores:

- $a(i)$: Mean distance between a sample i and all other points in the same cluster.
- $b(i)$: Mean distance between a sample i and all other points in the nearest cluster.

The coefficient for a single sample is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.1}$$

where $s(i)$ indicates well-defined clusters with higher scores.

The silhouette score measures how similar an object is to its own cluster compared to other clusters. A higher silhouette score indicates that objects are well-matched to their own cluster and poorly matched to neighboring clusters.

Cluster Validation Indices: To further validate the clustering configurations, the Calinski-Harabasz Index and Davies-Bouldin Index were employed:

Calinski-Harabasz Index[31]: Also known as the Variance Ratio Criterion, it is defined as:

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1}$$

where $\mathrm{Tr}(B_k)$ and $\mathrm{Tr}(W_k)$ represent the trace of the between-cluster and within-cluster dispersion matrices, respectively. A higher value generally indicates that the clusters are dense and well-separated, which is desirable.

Davies-Bouldin Index[32]: This index evaluates cluster separation by the average similarity between each cluster:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

The Davies-Bouldin Index is an internal evaluation scheme where lower values indicate better clustering.

Cluster Count for K-means **five** clusters were chosen as the optimal number to operate the k-means for several reasons:

- **Complexity and Variety:** The dataset contains complex and multidimensional data better represented through a more granular clustering approach, as evidenced by the significant secondary peak in the silhouette scores.
- **Cluster Validation:** Both the Calinski-Harabasz and Davies-Bouldin indices for five clusters suggest that this configuration offers a balanced approach, providing a clear separation and density of clusters, which is suitable for capturing the nuanced differences in musical features.
- **Practical Applicability:** In the context of a music recommendation system, having more clusters allows for a more nuanced categorization of music tracks, enhancing the personalization potential of the recommendations.

Therefore, based on a comprehensive assessment using multiple indices and considering the need for a detailed representation of the data's inherent diversity, five clusters were selected as the most appropriate configuration for our clustering model.

Enhancing Cluster Descriptions Enhancing the interpretability of clusters generated from KMeans clustering involved a detailed analytical process to assign descriptive labels that reflect the predominant characteristics of each cluster. This was achieved by examining and understanding each cluster's mean values of selected musical features.

Normalization and Cluster Description Generation: Initially, we calculated the mean values for key features such as acousticness, danceability, and energy for each cluster. We applied Z-score normalization to these means to determine which features significantly characterize each cluster. This normalization process calculates how many standard deviations a feature's mean value is from the overall mean across all clusters, thereby highlighting the most distinctive features for each cluster.

Generating Descriptive Labels Using Z-Scores Using the normalized data, we determined each cluster's three most important features by considering their absolute Z-scores. By choosing the top three features, we can guarantee that the labels accurately represent the most distinctive characteristics of the music songs in each cluster. This will result in clear and practical insights. Subsequently, these characteristics were transformed into simpler and user-centric descriptive terms to enhance comprehension and implementation. The listing shows the intuitive descriptive terms used for each feature in the cluster description:

```
# Define intuitive descriptions for features
feature_names_map = {
    'tempo': 'Varied Tempo', 'energy': 'Vibrant', 'valence': '
    Joyful',
    'danceability': 'Danceable', 'speechiness': 'Lyrical', '
    acousticness': 'Mellow',
    'instrumentalness': 'Instrument-rich', 'liveness': 'Live',
    'loudness': 'Loud'
}
```

Listing 3.1: Intuitive Descriptions for Features

Utilization of Cluster Descriptions The dataset was updated by adding a column called 'Cluster Description,' which included the labels based on the highest Z-scores. This column successfully conveys the core characteristics of each cluster using descriptive terms such as 'Vibrant / Danceable / Joyful', providing a clear and instant comprehension of the unique qualities that differentiate each cluster.

- Cluster 0 is described as *Varied Tempo / Vibrant / Mellow*, indicating a diverse range of tempos, vibrant energy levels, and a mellow tone.

- Cluster 1 is described as *Joyful / Danceable / Live*, highlighting its lively, upbeat, and dance-friendly characteristics.

- Cluster 2 features labels such as *Lyrical / Live / Danceable*, pointing to its strong vocal content, live performance feel, and suitability for dancing.

- Cluster 3 is captured with *Mellow / Vibrant / Joyful*, suggesting a blend of calm and energetic joyful music.

- Cluster 4 is marked by *Instrument-rich / Mellow / Danceable*, reflecting its instrumental richness and smooth and rhythmic dance quality.

These enhanced explanations help visualize and comprehend the data and play a crucial part in the recommendation system. These genre descriptions are now much simpler than the initial DB's features. Each song entitled with cluster and cluster description will ensure that the recommendation system can utilize content-based filtering for our recommendation system.

The dataset, now augmented with 'Cluster Description', has been saved as `data_final_with_descriptive_labels.csv` and is ready for subsequent use in developing the Large Language Model (LLM) and Hybrid recommendation models. This enhancement ensures that the recommendation systems can leverage these descriptive labels to offer highly personalized music recommendations, thus improving the overall user experience.

3.1.4 Merge Genre Data

We integrated and improved our dataset by combining comprehensive track metadata with genre information obtained from a secondary dataset[10]. Enriching the music files with genre data is a vital step that greatly enhances the performance of content-based filtering in our recommendation systems.

Data Merging Our primary dataset `data_final_with_descriptive_labels.csv` consists of a collection of 170,653 songs, including artist names and other musical attributes, whereas our secondary dataset [10] offers genre classification for 28,680 artists. **Left join** was performed to keep all the rows from our clustered described database (`data_final_with_descriptive_labels.csv`) and matches rows from the right genre dataset on the 'artists' column. Thorough preprocessing was necessary to guarantee that all artist names were consistent and correct. This entailed eliminating redundant whitespace and transforming lists of artist names into strings separated by commas.

Following the merging process, we included genre information from the second dataset in the merged data. Only about 1 in 5 of the songs (48,481 Tracks) did not have clear genre correlations and were categorized as 'Unknown' to maintain consistency in the data. In addition, we organized and eliminated duplicate genre entries for every artist, resulting in a concise and cohesive list of genres for each song. This phase is crucial for streamlining data handling and optimizing its usefulness in our recommendation system.

After merging, we retained only the necessary columns to ensure the dataset remained efficient and concentrated on important characteristics. The overview of the 3.2 is as following

#	Column	Non-Null Count	Dtype
0	artists	170653 non-null	object
1	name	170653 non-null	object
2	popularity	170653 non-null	int64
3	year	170653 non-null	int64
4	cluster	170653 non-null	int64
5	Cluster_Description	170653 non-null	object
6	genres	170653 non-null	object

Table 3.2: Summary of the dataset showing the column names, counts of non-null entries, and data types.

Storing the Improved Dataset The enhanced dataset, which includes genre and cluster descriptions, was saved as ‘`data_final_cleaned1.csv`‘. The dataset provides a solid basis for developing advanced hybrid recommendation systems. Utilizing the detailed musical characteristics and the recently included genre information, these systems may provide personalized and contextually appropriate music suggestions. Besides recommendation systems, the dataset may be utilized for other purposes, including market segmentation, target audience analysis, algorithmic music production, and academic music science studies.

3.1.5 Data Integration for Collaborative Filtering

Music streaming platforms have exponentially increased the volume of accessible music, creating challenges for users trying to discover new and appealing music and creating issues like long tails and cold starts. To tackle these issues, personalized music recommendation systems play a vital role. They assist consumers in navigating extensive music collections by recommending tracks that match their interests and preferences. Our research improves current systems by combining comprehensive track metadata with user interaction data from the Spotify Playlist Data [10], which includes important information such as user IDs, artist names, track titles, and playlist names. (`spotify_playlist.csv`) collected from Kaggle. The dataset can be accessed by visiting the following URL: https://www.kaggle.com/datasets/andrewmvd/spotify-playlists. This dataset captures real-time user interactions with music tracks, including data on users, their playlists, and the tracks within these playlists, which includes preprocessing steps similar to the initial datasets. These entries are essential to building a user interaction matrix and creating collaborative Filtering.

Significance of Spotify Playlist Data The Spotify Playlist Data is derived from users who share their *nowplaying* tracks through Spotify on social media platforms. This dataset captures real-time user interactions with music tracks, including data on users, their playlists, and the tracks within these playlists.

The Spotify Playlist Data [10] has a large collection of user interactions, totaling 12,891,680 items. The dataset is organized based on four primary columns.

- user_id: A unique identification assigned to each individual Spotify user.
- "artistname": the artist's name linked to a song.
- "trackname": The name given to the song.
- "playlistname": The designated title of the playlist in which the track is located.

Upon conducting a more in-depth examination of this dataset, the subsequent essential statistical information has been uncovered and summarized in Table 3.3

Statistic	**Value**
Total rows (interactions)	12,891,680
Total columns (features)	4
Non-null values:	
user_id	12,891,680
"artistname"	12,858,108
"trackname"	12,891,592
"playlistname"	12,890,434

Table 3.3: Key Statistics of the Spotify Playlist Data

Data Merging and Processing We merged this real-time user interaction data with our enriched track metadata from 'data_descriptive_labels.csv'. This integration was meticulously executed to ensure precision in matching track and artist names, thus reflecting exact user interactions with music tracks. Both datasets were normalized for accurate integration:

- Artist and track names were converted to lowercase and stripped of extra spaces to ensure uniformity.
- Artist names were split into lists to address multiple artist collaborations accurately.
- We employed the 'exploding' method to separate these lists into distinct rows, enhancing the granularity of our matching process.

The datasets were merged using an inner join based on artist and track names, ensuring only entries with exact matches were combined. This selective merging process ensures our dataset accurately represents genuine user interactions with the music.

3.1.6 Final Aggregated Dataset Preparation and Usage

Following the merge, we refined the dataset to focus on essential attributes, including user IDs, track names, artist names, playlist names, and other critical metadata. This streamlined dataset is crucial for constructing an accurate and efficient user interaction matrix. Then, following data integration, the dataset undergoes aggregation to analyze user interaction with tracks across various features such as playlists, clusters, and popularity. Key steps in the analysis include:

1. **Grouping Data:** Data is grouped by user, track, artist, and other relevant attributes. Each group's size, indicating the play count, is calculated to understand listening frequency.
2. **Validation:** The aggregated dataset is checked for missing values, particularly in key columns, to ensure the integrity of the analysis.
3. **Data Saving:** The cleaned and aggregated data is saved as '`aggregrated_data_full.csv`' for easy access and further analysis.

The fully integrated and refined saved dataset now contains 4,419,267 entries, each enriched with comprehensive metadata and user interaction details. This dataset forms the backbone of our user interaction matrix, crucial for analyzing user behavior and enhancing the recommendation algorithms. The structure 3.4 and sample table 3.5 is as follows:

Column	Data Type	Description
user_id	object	Unique identifier for each user
trackname	object	Name of the track
artistname	object	Name of the artist of the track
playlistname	object	Name of the playlist the track belongs to
cluster	int64	Cluster no based on track attributes
Cluster_Description	object	Description of the cluster the track belongs to
year	int64	Year the track was released
popularity	int64	Popularity score of the track
genres	object	List of genres associated with the track
play_count	int64	Number of times a track has been played by a user

Table 3.4: Fully Aggregated Dataset Information

UID	Track Name	Artist Name	Playlist Name	Cluster	Year	Popularity
064e	Love The Way You Lie	Eminem	Starred	1	2010	82
ee0	Team	Lorde	2014	1	2013	72
e80	No Sleep	Wiz Khalifa	30m cardio	0	2011	66
0e2	God's Gonna Cut You Down	Johnny Cash	Old classics	1	2006	64
3f4	Resolution	Matt Corby	Triple j's Hottest 100 2013	0	2013	48

Table 3.5: A selection of records from the fully aggregated dataset showcasing diverse music preferences and playback frequencies.

Now this ‘`aggregrated_data_full.csv`‘ is ready for deployment within our recommendation system. It contains features necessary to handle the traditional recommendation system issues like long tail and cold start. Effectively utilizing these features we aim to build our recommendation system for a better musical experience

3.2 Development of Recommendation Systems

This section outlines the development of advanced music recommendation systems that leverage the `Aggregated_DB` and its detailed user interaction data. These systems are developed to address specific challenges like cold start and long tail problems, along with capturing the semantic mood of the user. Three recommendation systems were built using a sophisticated integration of collaborative, content-based, and context-aware filtering techniques.

1. **Playlist Recommendation System: Collaborative Filtering**

   (a) **Constructing User Interaction Matrix:** Utilizing the `Aggregated.csv`, we construct a user interaction matrix based on play count per playlist, ensuring a robust foundation for collaborative filtering.

   (b) **Calculating Similarity Scores:** User validation and similarity ratings are computed to determine the closeness of user preferences, enhancing the relevance of recommendations.

   (c) **Scoring and Recommendations:** Each playlist is scored by calculating relevance based on the interactions of similar users. The system then recommends playlists excluding those the user has previously interacted with, thus providing fresh and relevant content.

2. **Hybrid Song Recommendation System: Content-Collaborative Filtering**

   (a) **Interaction Matrix Construction:** For songs, an interaction matrix is built based on play counts per song from `Aggregated.csv`, formatted into a dense matrix for efficient processing.

   (b) **Applying Dimensionality Reduction:** Truncated SVD is applied to reduce the dimensionality of the interaction matrix, facilitating more efficient and precise collaborative filtering.

   (c) **Hybrid Filtering Application:** The system integrates content-based filtering (genre and artist matching) and collaborative filtering (similarity between songs and interaction weight) to generate personalized song recommendations based on user input.

3. **Context-Aware Query-Based LLM Filtering System:**

   (a) **Query Processing and Mapping:** Processes user prompts (e.g., "Rock Mix 20") to map out relevant genres, moods, and decades.

(b) **Multi-dimensional Filtering:** Applies content-based filtering to match genres and moods and context-based filtering for decade relevance.

(c) **Sentiment Analysis Integration:** Incorporates a DistilBERT-based LLM to analyze and adjust recommendations according to the emotional tone or sentiment of the user's input.

(d) **Final Recommendations:** Delivers highly personalized music suggestions based on an amalgamation of content, context, and emotional analysis.

This comprehensive approach not only overcomes semantic barriers and enhances user engagement but also effectively addresses challenges related to new user integration and the discovery of less popular, niche content, ensuring a robust and inclusive music recommendation experience for all kinds of music lovers.

3.2.1 Playlist Recommendation System: Collaborative Filtering

The playlist recommendation system we use utilizes advanced collaborative filtering processes to offer tailored choices for playlists. This method employs playlist user interaction data to recommend new material that aligns with user preferences.

As Figure 3.3 demonstrates, flow to build the collaborative playlist recommendation system.

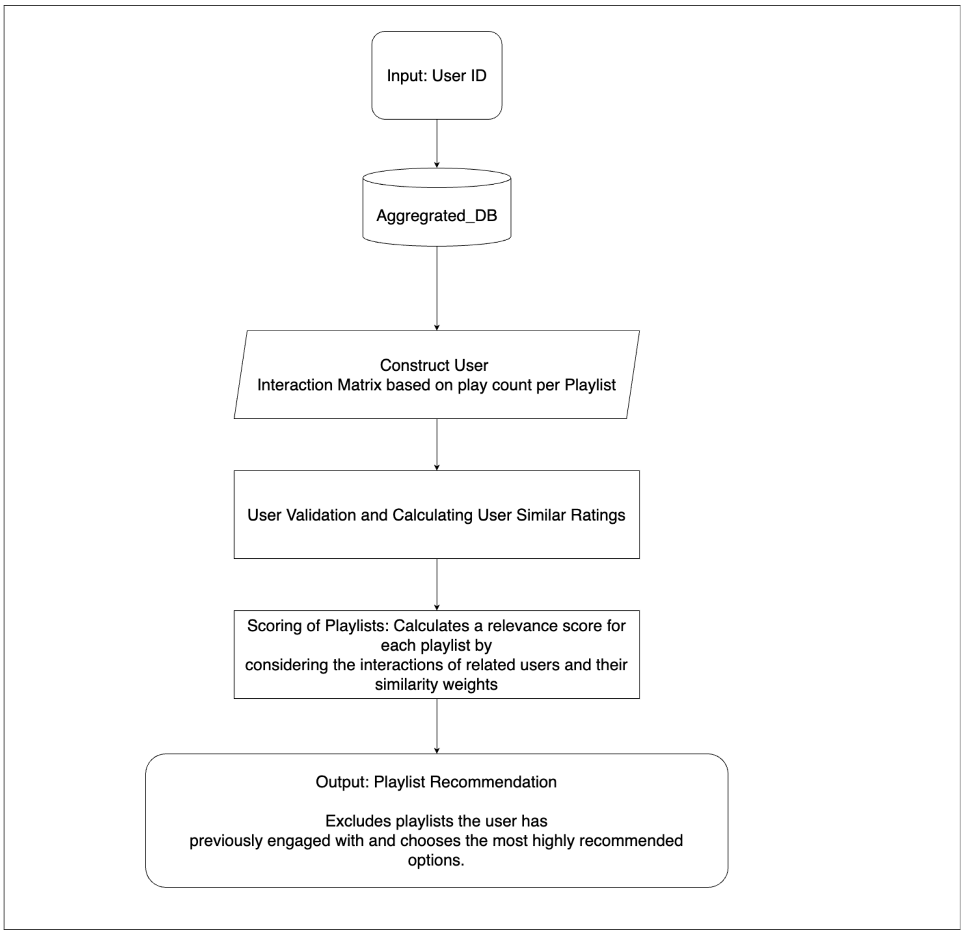

Input: User ID
Aggregrated_DB
Construct User Interaction Matrix based on play count per Playlist
User Validation and Calculating User Similar Ratings
Scoring of Playlists: Calculates a relevance score for each playlist by considering the interactions of related users and their similarity weights
Output: Playlist Recommendation
Excludes playlists the user has previously engaged with and chooses the most highly recommended options.

Figure 3.3: Collaborative filtering for playlist recommendations

Playlist Interaction Matrix : We have created a matrix for playlist interaction to improve our comprehension and utilization of the combined data. In this matrix, each row corresponds to a user, and each column corresponds to a playlist. This matrix is crucial for showing the wide variety of user interactions, even if sparse, because of user engagement's many playlists and selective nature.

Data and Preprocessing The cornerstone of our recommendation system is the interaction matrix, in which the rows represent users, and the columns represent playlists. The matrix is effectively saved sparsely using the NPZ file format, reducing storage space and improving load times. Users and playlists are efficiently mapped during the recommendation process using dictionaries for indexing.

```
# Load user and playlist mapping dictionaries
with open(f'{dataset_path}/user_id_to_index.pkl', 'rb') as f:
    user_id_to_index = pickle.load(f)
with open(f'{dataset_path}/index_to_playlist_name.pkl', 'rb')
   as f:
    playlist_index_to_name = pickle.load(f)
```

Listing 3.2: Python script to load user and playlist mapping dictionaries.

Methodology The `recommend_playlists` function is designed to operate efficiently, even when dealing with extensive data sets. Below is a detailed analysis of the process of the function:

1. **User Validation:** Verifies the presence of the user ID in our collection, addressing scenarios where the user may be new or missing.

2. **Calculation of Similarity:** Utilizes user similarity ratings to locate individuals with comparable musical preferences.

3. **Scoring of Playlists:** Calculates a relevance score for each playlist by considering the interactions of related users and their similarity weights.

4. **Compilation of Recommendations:** Excludes playlists the user has previously engaged with and chooses the most highly recommended options.

```
def recommend_playlists(user_id, user_similarity_matrix,
   interaction_matrix,
                            user_id_to_index,
   playlist_index_to_name, top_n=10):
    user_index = user_id_to_index.get(user_id)
    if user_index is None:
        return "User ID not found."
    similarity_scores = user_similarity_matrix[user_index]
    similar_indices = np.argsort(-similarity_scores)[:top_n+1]
    playlist_scores = np.dot(similarity_scores[similar_indices
   ],
                                 interaction_matrix[similar_indices
   ].toarray())
    recommended_playlists = [playlist_index_to_name[idx]
                             for idx in np.argsort(-
   playlist_scores)[:top_n]]
    return recommended_playlists
```

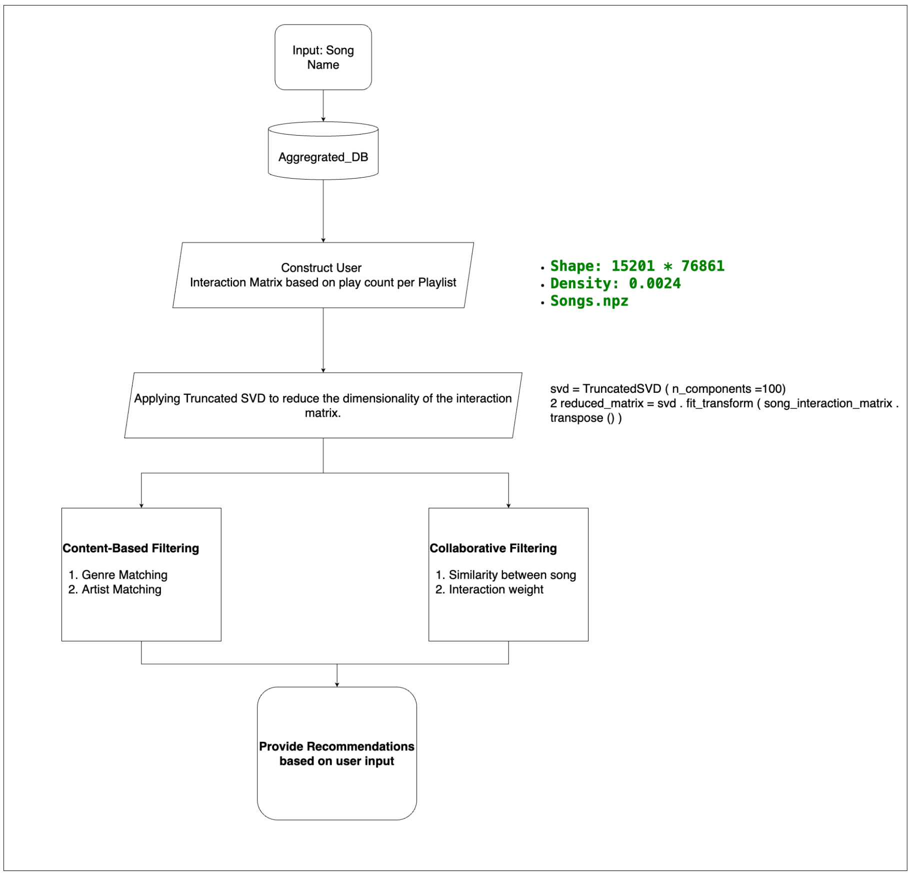
Listing 3.3: Function to recommend playlists based on user similarity.

3.2.2 Content-Collaborative Song Recommendation System

Our hybrid music recommendation system utilizes advanced **collaborative** and **content-based filtering** approaches to deliver tailored song recommendations.

This strategy combines the advantages of both strategies by utilizing user interaction data and integrating music metadata to enhance the accuracy and relevance of recommendations.

As Figure 3.4 illustrates, flow to build the hybrid collaborative and content-based song recommendation system.


Input: Song Name
Aggregrated_DB
Construct User Interaction Matrix based on play count per Playlist
• Shape: 15201 * 76861
• Density: 0.0024
• Songs.npz
Applying Truncated SVD to reduce the dimensionality of the interaction matrix.
Content-Based Filtering
1. Genre Matching
2. Artist Matching
Collaborative Filtering
1. Similarity between song
2. Interaction weight
Provide Recommendations based on user input

```
svd = TruncatedSVD ( n_components =100)
reduced_matrix = svd . fit_transform ( song_interaction_matrix .
transpose () )
```

Figure 3.4: Hybrid filtering for Song recommendations

User Interaction Matrix: To improve our comprehension and use of the combined data, we have created an interaction matrix in which rows and tracks represent individuals represented by columns. This matrix is crucial for illustrating the wide range of user involvement, even if sparse, due to the diverse selection of tracks and the selective nature of user interactions.

Data Preprocessing and Reducing Dimensionality Our recommendation method is based on a strong interaction matrix and extensive music metadata. This structure is crucial for carefully evaluating user preferences and song features. The sparse style of the interaction matrix efficiently stores and optimizes both storage space and computing performance, effectively expressing user interactions with music.

```python
# Load the aggregated song data
song_data = pd.read_csv(f'{dataset_path}/aggregated_data_full.
    csv')
song_data['trackname'] = song_data['trackname'].str.lower().str
    .strip()
song_data.drop_duplicates(subset=['trackname'], inplace=True)
song_data.reset_index(drop=True, inplace=True)
song_data['trackname'] = pd.Categorical(song_data['trackname'])
song_name_to_index = {name: index for index, name in enumerate(
    song_data['trackname'].cat.categories)}
song_interaction_matrix = load_npz(f'{dataset_path}/
    sparse_song_matrix.npz')
```

Listing 3.4: Python script to preprocess and load song data and the interaction matrix.

To address the problems caused by having many dimensions and improve the speed of similarity calculations, we utilize Truncated Singular Value Decomposition (SVD) on the transposed version of the interaction matrix. This process aids in diminishing noise and computational intricacy while maintaining the fundamental attributes of the data.

```python
svd = TruncatedSVD(n_components=100)
reduced_matrix = svd.fit_transform(song_interaction_matrix.
    transpose())
```

Listing 3.5: Applying Truncated SVD to reduce the dimensionality of the interaction matrix.

Methodology The hybrid recommendation function is an advanced combination of collaborative and content-based filtering components, specifically developed to enhance the song suggestion process by leveraging user interaction data and song information.

Content-Based Filtering: This aspect of the algorithm improves the quality of recommendations by utilizing comprehensive metadata about music. These criteria affect the calculation of similarity scores, ensuring that songs with common content features are given higher priority:

- **Matching genres:** If the genres of a candidate song align with the genres of the target song, the score is substantially augmented. This illustrates the fundamental assumption that consumers prefer songs that belong to the same or related genres.

- **Consideration of metadata:** The program also considers other metadata, such as the artist's resemblance and the release year of the recordings.

This process enhances the suggestions by adjusting them to match user preferences and current popular preferences better.

Collaborative Filtering: This section employs patterns from user interactions to discern preferences among users with similar listening habits.

- **Scores indicating the similarity between users:** A precomputed similarity matrix reveals users whose preferences closely match the target user. These scores are utilized to modify the impact of music these individuals favor.

- **Weights for Interaction:** Tracks with greater relevance ratings are those that are preferred by users with similar musical preferences, indicating that these songs are likely to be appealing to the target user as well.

Improving Scores and Combining Them The recommendation scores are calculated by using insights from both content-based and collaborative filtering methods. These scores are then dynamically adjusted using direct content matches and inferred user preferences.

- Calculation of Initial Score: The scores are calculated based on the cosine similarity between the feature vectors of the songs, obtained from a matrix with decreased dimensionality. This measure quantifies the degree of similarity between each song and the target song in the reduced feature space.

- **Improvements based on content:** Subsequently, the scores are modified to represent the material's similarity accurately. The following is a list item: Songs that belong to the same genres as the target song will have their score doubled, indicating a strong similarity in terms of substance. If songs belong to the same cluster, their score is augmented by 50%, highlighting the structural similarity within the data.

- Collective modifications: The initial scores are enhanced by considering collaborative signals. If songs have the same popularity level, a little rise of 10% is applied, showing that users generally agree on their attractiveness. Scoring is reduced proportionately to the gap in release years, considering its decreasing relevance over time. Less importance is assigned to songs much older or fresher than the target song.

```python
def hybrid_recommend_songs(song_name, song_name_to_index,
    reduced_matrix, song_data, top_n=5):
    # Extract song index from the name to index map
    song_index = song_name_to_index.get(song_name.lower().strip
    ())
    if song_index is None:
        return [("Song not found.", "", "", 0)]

    # Compute cosine similarity scores from the reduced SVD matrix
    similarity_scores = cosine_similarity([reduced_matrix[song_index]], reduced_matrix)[0]

    # Enhance scores based on genre and other metadata similarities
    enhanced_scores = []
    target_genres = set(song_data.at[song_index, 'genres'].split(','))
    for idx, score in enumerate(similarity_scores):
        if idx != song_index and idx < len(song_data):
            song_meta = song_data.iloc[idx]
            score_adjustment = score * 1.5 if song_meta['cluster'] == song_data.at[song_index, 'cluster'] else score
            if set(song_meta['genres'].split(',')) & target_genres:
                score_adjustment *= 2  # Boost score for genre matches
            enhanced_scores.append((score_adjustment, idx))

    # Select top N recommendations after sorting by adjusted scores
    recommended_indices = sorted(enhanced_scores, reverse=True, key=lambda x: x[0])[:top_n]
    recommended_songs = [(song_data.iloc[idx]['trackname'], song_data.iloc[idx]['artistname'], song_data.iloc[idx]['genres'], score) for score, idx in recommended_indices]
    return recommended_songs
```

Listing 3.6: Function to compute hybrid recommendations combining collaborative and content-based signals, reflecting both user behavior and song characteristics.

The extensive scoring procedure guarantees that users favor each suggested song with similar tastes and closely match the target user's exact content preferences. This enhances the customized experience and pleasure with the recommended music.

3.2.3 Advanced Query-Based Song Recommendation System

Our LLM-based hybrid music recommendation system combines the powerful features of collaborative and content-based filtering with the deep comprehension of natural language context offered by the transformers library. Our solution surpasses conventional models by integrating sentiment analysis and contextual data interpretation, enabling context-based suggestions. Leveraging LLM model named the distilbert-base-uncased-finetuned-sst-2-english[33], our method employs user interaction data, comprehensive song metadata, and the attitude expressed in user inquiries to offer highly individualized and relevant music recommendations.

As Figure 3.5 demonstrates, flow to build the advanced LLM-based hybrid System.

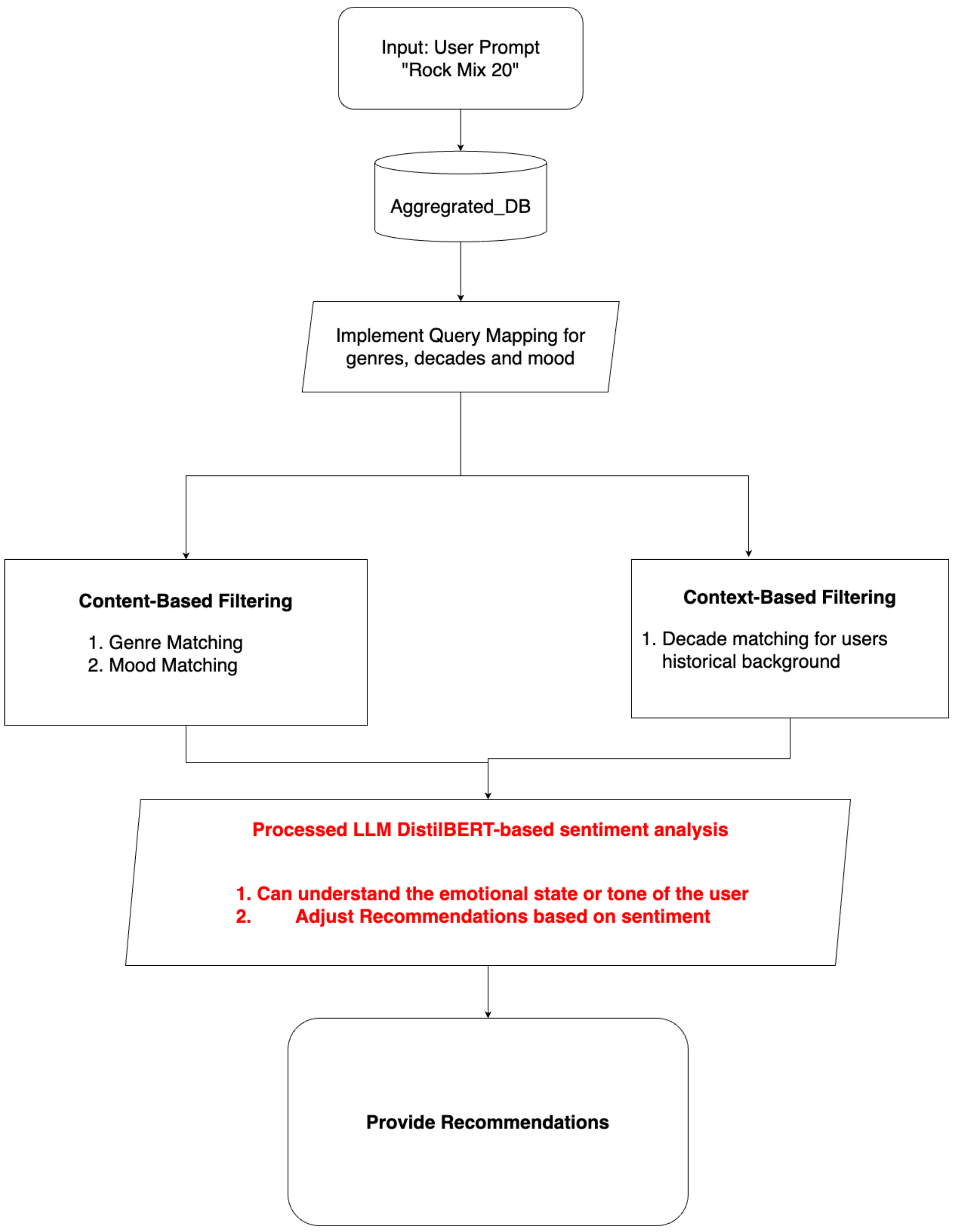

Input: User Prompt "Rock Mix 20"
Aggregrated_DB
Implement Query Mapping for genres, decades and mood
Content-Based Filtering
1. Genre Matching
2. Mood Matching
Context-Based Filtering
1. Decade matching for users historical background
Processed LLM DistilBERT-based sentiment analysis
1. Can understand the emotional state or tone of the user
2. Adjust Recommendations based on sentiment
Provide Recommendations

Figure 3.5: Hybrid filtering for Song recommendations

Mapping of Genre, Mood, and Decade Using a combination of content-based, context-based, and collaborative filtering approaches, our music recommendation engine carefully analyzes user queries to extract important music features. With the help of this diverse approach, the system can offer highly customized music recommendations that are emotionally and contextually in line with the user's interests.

Implementing Query Mapping: In order to interpret user input and discover relevant musical properties, such genres, moods, and favored decades, the extraction process requires many important operations. Below is a comprehensive examination of the code implementation:

```python
import re

# Generates keywords associated with each decade, useful for filtering songs by year
def generate_decade_keywords():
    temporal_keywords = {}
    for year in range(1920, 2030, 10):
        decade_key = f"{year // 10 * 10}s"
        temporal_keywords[decade_key] = [str(year // 10 * 10), f"{str(year // 10 * 10)[2:]}s", decade_key]
    return temporal_keywords

# Maps user queries to genres, moods, and decades based on predefined keyword lists
def map_query_to_genres_moods_and_decades(query):
    query = query.lower()
    found_genres, found_moods, found_years = set(), set(), set()

    # Genre keywords are mapped to various music styles
    genre_keywords = {
        'rock': ['rock', 'punk', 'grunge', 'alternative'],
        'pop': ['pop', 'synthpop', 'electropop'],
        # More genres...
    }
    # Mood keywords capture the emotional content of music
    mood_keywords = {
        'happy': ['joyful', 'cheerful', 'uplifting'],
        'sad': ['melancholic', 'sombre', 'downtempo'],
        # More moods...
    }
    temporal_keywords = generate_decade_keywords()

    # Identifying genres in the query
    for genre, keywords in genre_keywords.items():
        if any(keyword in query for keyword in keywords):
            found_genres.add(genre)
    # Identifying moods in the query
    for mood, keywords in mood_keywords.items():
        if any(keyword in query for keyword in keywords):
            found_moods.add(mood)
    # Matching decades based on keywords
    for decade, keywords in temporal_keywords.items():
        if any(keyword in query for keyword in keywords):
            start_year = int(decade[:-1])
            found_years.update(range(start_year, start_year + 10))
    return found_genres, found_moods, found_years
```

Listing 3.7: Mapping user queries to music characteristics

Function in the Recommendation System: The mapping process is crucial for several reasons:

- **Content-based filtering:** The algorithm can search the music library for songs that fit the indicated qualities by extracting genres and moods from user queries. This allows the system to identify songs that are specifically tailored to the user's likes.

- **Contextual Filtering:** The algorithm may recommend music that matches current trends and connects with the user's desired historical background by recognizing phrases connected to the decade.

By combining these techniques, recommendations are made that are significantly customized and relevant, responding to the user's implicit emotional cues and explicit demands. This advanced methodology promotes a more engaging and satisfying user experience by offering music suggestions that connect with personal tastes.

LLM Model Integration Using language models to improve sentiment analysis, our music recommendation system includes the distilbert-base-uncased-finetuned-sst-2-english model from Hugging Face's transformers library[33]. This model is a reduced variant of the broader BERT model, designed exclusively for sentiment analysis. It offers a lightweight yet very effective tool for reading user emotions from text.

- **Execution of Sentiment Analysis:** The sentiment analysis model is integral to parsing the emotional content of user queries. It allows our system to comprehend not only the explicit preferences users express but also the implicit emotional undertones that are crucial for tailoring music suggestions. Here is how we use this model to adjust the dynamics of our recommendations:

```python
from transformers import pipeline

# Load the sentiment analysis model
sentiment_analyzer = pipeline("sentiment-analysis", model=
    "distilbert-base-uncased-finetuned-sst-2-english")

def analyze_sentiment(query):
    # Obtain sentiment prediction
    result = sentiment_analyzer(query)
    score = result[0]['score']
    return score if result[0]['label'] == 'POSITIVE' else
   -score

# Example usage within the recommendation system
query = "I need uplifting music to boost my mood."
sentiment_score = analyze_sentiment(query)

```

Listing 3.8: Sentiment analysis in recommendation system

- **Contextual Response:** The system not only considers the genres or musicians mentioned but also adapts its recommendations to match the mood suggested by the user's language, guaranteeing that the recommended music connects with the user on an emotional level. Examining the Constraints and Possibilities of the Addressing System Although DistilBERT successfully captures a wide range of moods, it has inherent limitations because it is trained on generic datasets instead of music-specific text. Although it functions well given the computing limitations of our current configuration, there is potential for enhancement.

- **Sensitivity of the model:** Future improvements might involve retraining or fine-tuning the model using music-specific datasets, strengthening its ability to comprehend the subtleties unique to musical mood and context.

- **computing Efficiency:** Although DistilBERT is a less resource-intensive model, it still demands substantial computing resources, particularly when expanding to accommodate a high volume of users in real time. Implementing optimizations or adopting even more efficient models might help mitigate these issues. Potential Avenues for Future Development to enhance our sentiment analysis skills, we may investigate the following:

  - **Advanced Models:** Employing more recent or highly specialized models might improve the precision and comprehensiveness of sentiment analysis.

  - **User Feedback Loop:** Incorporating user feedback into the training process enables the model to adapt its comprehension in response to real-world user interactions and preferences.

Integrating DistilBERT into our music recommendation algorithm greatly improves our capacity to provide tailored suggestions. Our technology offers personalized music recommendations that are specifically matched to users' stylistic tastes and emotional states. These suggestions are carefully connected with the user's context and emotional demands. As we consider the future, our goal is to fully use advancements in language processing to enhance the user experience consistently.

Recommendation System Logic Our algorithm, which combines sentiment analysis and processed data from user queries to produce precisely matched music recommendations, is the brains behind our music recommendation system. This algorithm utilizes collaborative, content-based, and context-based filtering techniques to offer recommendations that are not just pertinent but highly customized.

Combining Sentiment Analysis and Data Processing: The recommendation engine improves its options by using sentiment ratings and categorizing genres, moods, and decades obtained from user inquiries. Each component plays a role in the recommendation process:

```python
def recommend_songs(song_data, genres, moods, decades, sentiment_score):
    # Filter songs by genres, moods, and decades extracted from the user's query
    filtered_songs = song_data[
        song_data['genres'].apply(lambda g: any(gen in g for gen in genres)) &
        song_data['moods'].apply(lambda m: any(mood in m for mood in moods)) &
        song_data['decades'].apply(lambda d: d in decades)
    ]

    # Adjust the popularity score based on the sentiment score to reflect the emotional context
    filtered_songs['adjusted_score'] = filtered_songs['popularity'] * (1 + sentiment_score)
    # Sort songs by the new adjusted score to get the top recommendations
    recommended_songs = filtered_songs.sort_values(by='adjusted_score', ascending=False).head(10)
    return recommended_songs
```

Listing 3.9: Combining sentiment analysis with genre, mood, and decade mappings to generate recommendations

Functionality and Impact:

1. **Content-based Filtering:** The engine guarantees that the recommendations closely fit the user's current tastes by matching music to the provided genres and moods in the query.

2. **Contextual Filtering:** Decade mapping enables the system to suggest songs from particular time periods, accommodating the user's nostalgic inclinations or facilitating the exploration of new musical eras.

3. **Sentiment Analysis:** The recommendation intensity is modified by integrating sentiment scores. Positive emotions might enhance the inclination towards vibrant and energetic tunes, while negative thoughts may suggest more calming or sad tracks.

Recommendation selection using ranking: At first, songs are sorted according to the genres, moods, and decades derived from the user's query to guarantee they match the user's stated tastes. Afterward, the popularity score of each song is adjusted dynamically to include the sentiment score, which is a numerical representation of the emotional content of the user's input.

1. **Adjustment of the base popularity:** Every song's baseline popularity reveals its overall level of appeal. The score is adjusted based on the sentiment score derived from the sentiment analysis of the query.

2. **Formula for Adjustment:** The formula employed in our system is as follows:

$$adjusted\_score = popularity \times (1 + sentiment\_score) \tag{3.2}$$

3. **user's present emotional condition:** Happy sentiment directly correlates with an increase in the popularity score, prioritizing songs that amplify happy sentiment and decreasing negative attitude impacts the score, indicating music that may provide relief or inspire thought.

4. **Ranking and Selection Results:** These modified scores are used to rank the songs, and the top-ranked songs are chosen as suggestions. This technique guarantees that recommendations are customized based on musical tastes and emotional situations.

Our recommendation engine creates a personalized listening experience by constantly modifying suggestions depending on these complex inputs, providing more than music recommendations. It considers the user's mood, preferences, and past interests. This guarantees that every suggestion list is precise in terms of musical content and emotionally impactful, offering customers a profoundly gratifying listening experience that feels individually tailored.

This methodical and comprehensive approach fulfills the user's particular requirements and improves the listening experience by harmonizing it with their emotional state and cultural background, demonstrating the effectiveness of a really adaptable music recommendation system.

4 Results

This section presents a detailed overview of the results derived from the methodologies implemented in this thesis, as shown in the flowchart in Figure **??**. The results are divided into two main subsections: one focusing on the outcomes of the data processing techniques and another detailing the performance of the recommendation systems. Each subsection critically evaluates the respective approaches, demonstrating their impact on enhancing the effectiveness of building a good recommendation system for music listeners.

1. **Data Processing Results:** This section will examine the initial data handling and feature extraction stages, assessing how well the preprocessing steps prepared the datasets for effective clustering and subsequent recommendation tasks.

2. **Recommendation System Results:** Following the data processing review, this subsection will focus on the operational performance of the Hybrid and Query-Based Recommendation Systems. It will explore the systems' accuracy, user engagement, and satisfaction rates, emphasizing how they address the challenges of cold start and long tail scenarios in music recommendation.

4.1 Data Processing Results

4.1.1 Feature Analysis and Dimensionality Reduction

Exploratory Data Analysis EDA showcased the distribution characteristics of various musical features, as illustrated in Figure 4.1. Each subplot represents the distribution of a different musical feature within the dataset. A Kernel Density Estimate (KDE) overlaid on each histogram provides a smooth estimate of the underlying probability density function. The KDE helps visualize the shape of the distribution more clearly, allowing for easier identification of patterns such as skewness, modality, and the central tendency of the data. This enhancement is crucial for understanding musical features' density and distribution characteristics, facilitating deeper insights that inform the development of the content-based recommendation system.

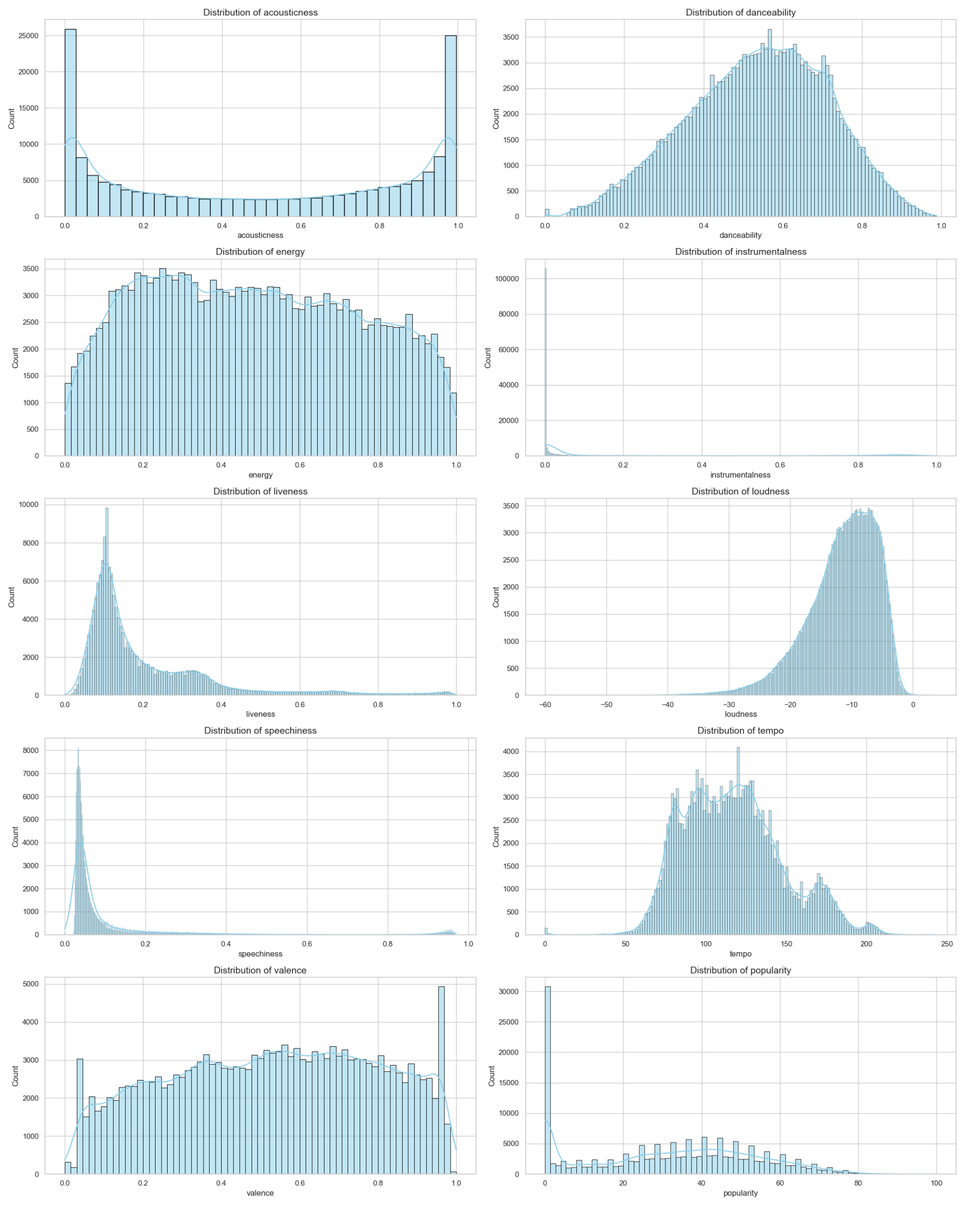


Figure 4.1: Histograms showing the distribution of various musical features within the dataset. Each subplot represents the distribution for a different feature, with 'kde' overlaid to indicate the density estimation.

Features such as 'danceability' and 'energy' exhibit approximately normal distributions, indicating a balanced representation across tracks. Conversely, 'acousticness', 'instrumentalness', 'liveness', and 'speechiness' show right-skewed distributions, suggesting a concentration of tracks with lower values in these attributes. 'Loudness' is left-skewed, reflecting a propensity for tracks to have higher volume levels. The ’valence’ feature presents a more uniform distribution, indicating a wide range of emotional expressions in the music. ’Tempo’ follows a pseudo-normal distribution with a slight right skew, and ’popularity’ displays a bimodal distribution, highlighting varying levels of listener engagement across tracks. These distributions provide invaluable insights into the dataset’s composition, which is crucial for developing an effective recommendation system.

Correlation Matrix Insights from the Correlation Matrix The heatmap can reveal several noteworthy correlations among our features. A detailed correlation analysis among musical features was conducted to uncover the relationships between different attributes. The correlation matrix, illustrated in Figure 4.2, highlights several noteworthy relationships:

- A strong positive correlation between ’energy’ and ’loudness’, suggesting that tracks with higher energy levels typically exhibit higher loudness.
- A significant negative correlation between ’acousticness’ and ’energy’, indicating that more acoustic tracks tend to have lower energy levels.
- ’Danceability’ shows a moderate positive correlation with ’valence,’ supporting that more danceable tracks often carry a happier or more positive mood.

These correlations provide insights into how various musical elements interact and can be clustered to provide more personalized songs to the user. These insights are crucial for the sophisticated algorithms used in the recommendation system.

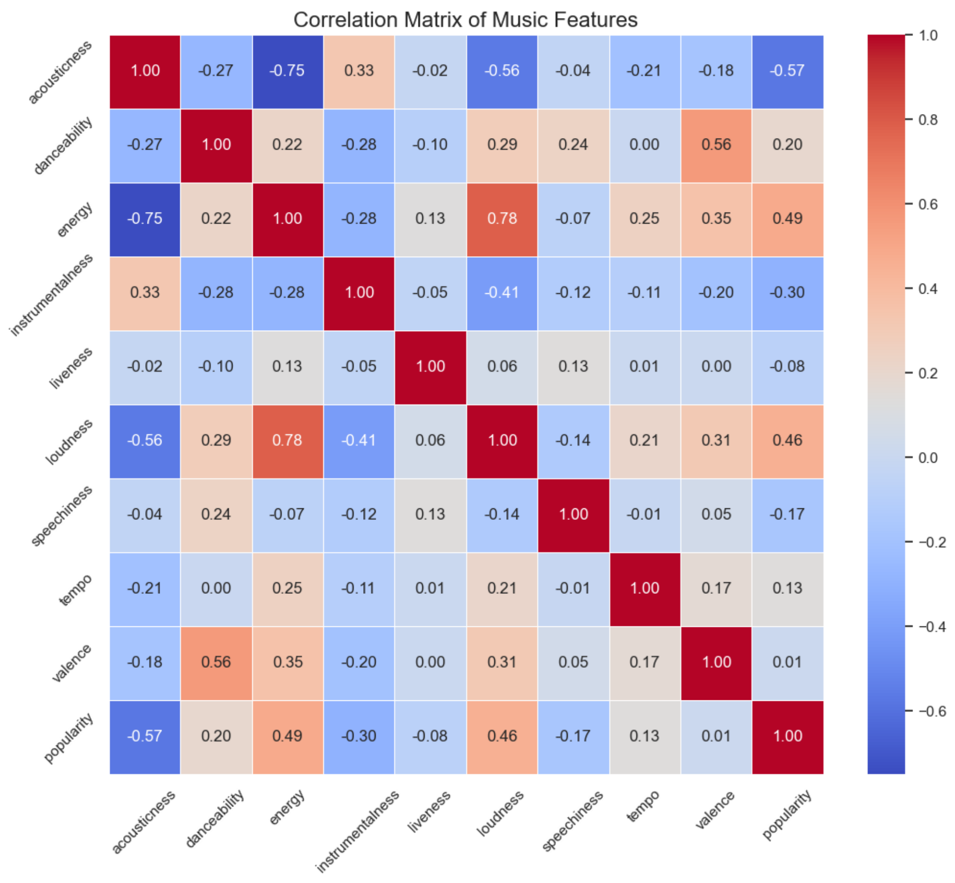


Figure 4.2: Correlation matrix of musical features, depicting the relationship between different attributes. Higher positive values (red) indicate a strong positive correlation, while deeper blue tones suggest negative correlations.

4.1.2 K-Means Clustering

Optimal Cluster Count Determination The silhouette scores for different numbers of clusters from 2 to 10 were calculated and are illustrated in Figure 4.3.

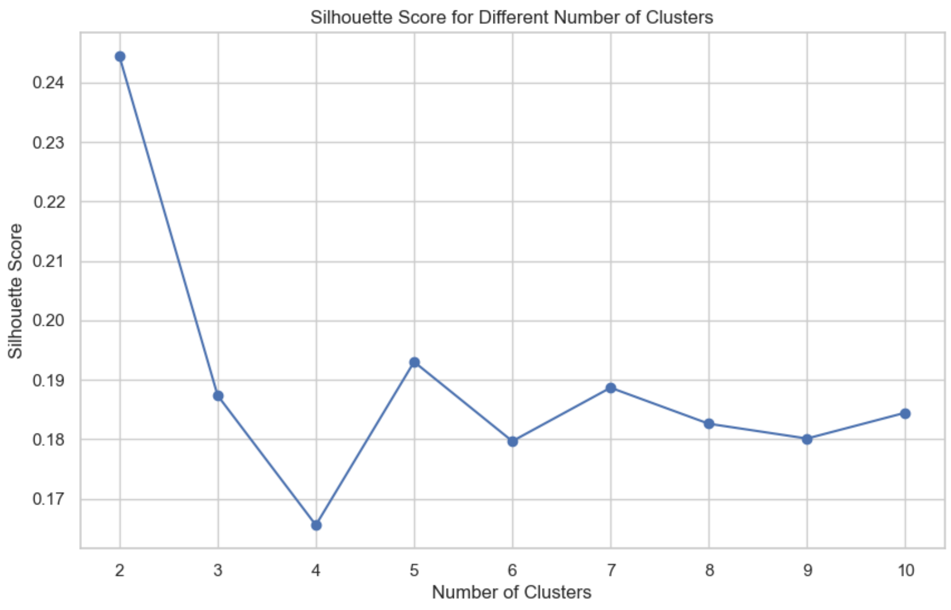


Figure 4.3: Plot of silhouette scores for different numbers of clusters, indicating the quality of cluster separation.

The silhouette analysis revealed two noteworthy peaks: the highest at two clusters with a silhouette score of 0.244 and another significant peak at five clusters with a score of 0.193.

For the final configuration of 5 clusters, the **CHI** index was calculated to be **31732.59**, which is reasonably high.

The **DBI** index for 5 clusters was 1.427, suggesting a reasonable configuration where clusters are neither too dispersed nor overlapping.

Clustering We performed KMeans clustering to organize the dataset into five groups, chosen based on the silhouette analysis that suggested this number strikes the best balance for our data. The clustering process was consistent, utilizing a fixed random state to ensure our results were reproducible. we analyzed each cluster by computing the mean of the original features for each group. Each piece of music in our dataset was assigned a cluster label, grouping it with similar tracks. The cluster labels now consist of count as follows:

- Cluster 0: 38,491
- Cluster 1: 54,748

- Cluster 2: 5,687
- Cluster 3: 47,488
- Cluster 4: 24,239

Visualizing Clusters:

Fig 4.4 illustrates the arrangement of the data points across the first two principal components and the position of each cluster's centroid. Different colors and red 'x represent different clusters' marks indicate the centroid of each cluster.

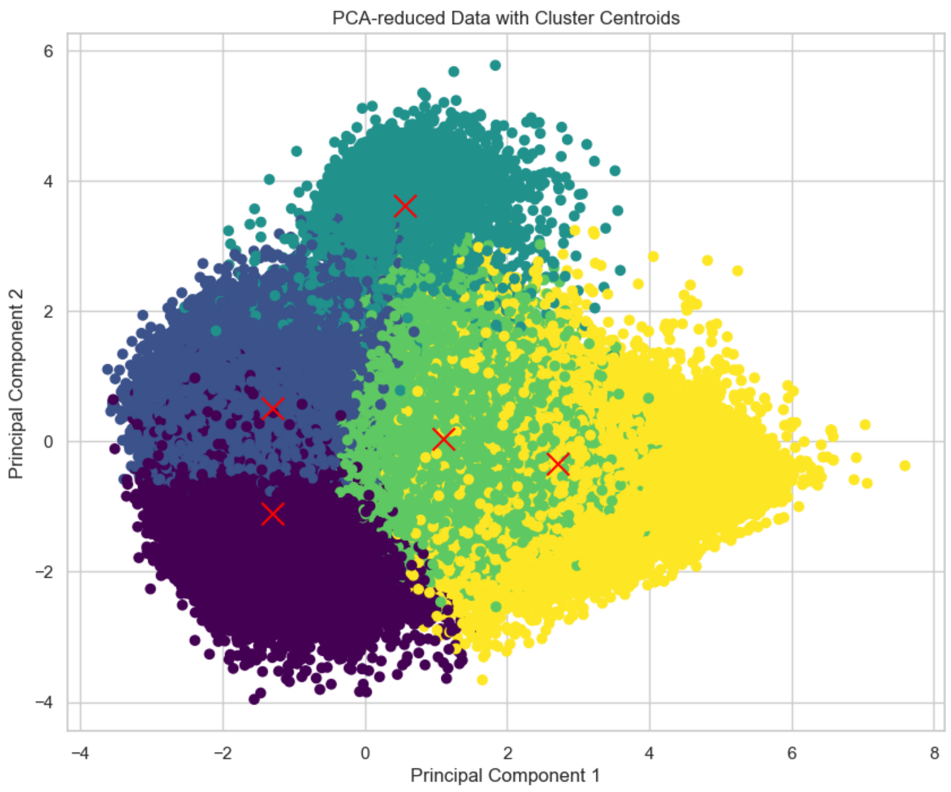


Figure 4.4: Visualization of KMeans clusters in PCA-reduced 2D space with cluster centroids marked, showing the separation and grouping of data points.

The diagram illustrates the distribution and intersection of the clusters in the reduced feature space. The visualization utilizes Principal Component 1 as the horizontal axis and Principal Component 2 as the vertical axis. These axes offer a clear understanding of the grouping of data points and their relationship to the cluster centroids.

Analysis The scatter plot demonstrates a degree of overlap between clusters, suggesting regions where the distinction between different music groups is not as well-defined. The close closeness of centroids indicates the presence of commonalities among certain clusters. Nevertheless, the clear patterns in the plot validate our decision to use six clusters, enabling a refined categorization that enhances the ability to provide a wide range of music suggestions.

The representation of clusters in the feature space reduced using PCA confirms the consistency of the produced clusters and highlights the complex nature of categorizing music data. The graphical representation of our music data's underlying structure is intriguing and supports our clustering strategy for a recommendation system that desires to represent a wide musical variety.

Also, a three-dimensional scatter plot was generated to explore the cluster distribution across the first three principal components, offering a deeper view of the data's underlying structure.

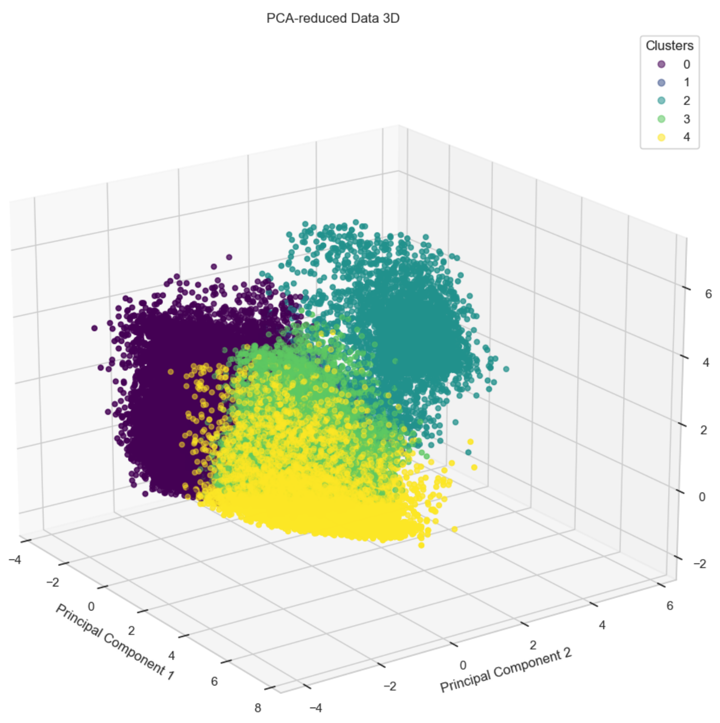


Figure 4.5: 3D visualization of KMeans clusters in PCA-reduced space, enhancing the perspective on spatial distribution and inter-cluster distances.

These visualizations corroborate our numerical analysis and provide intuitive graphical representations that aid in understanding the complex multidimensional nature of data clustering.

4.2 Recommendation Model Results

4.2.1 Collaborative Filtering for Playlist Recommendation System

When the recommendation function is executed with a certain user ID, the system generates a list of the ten most highly suggested playlists. These suggestions are generated by evaluating trends in user activity and matching them with similar profiles.

```
# Example of generating recommendations
recommended_playlists = recommend_playlists(example_user_id,

    user_similarity_matrix,

    playlist_interaction_matrix,
                                                      user_id_to_index,

    playlist_index_to_name)
print("Recommended Playlists:", recommended_playlists)
```

Listing 4.1: Example of generating playlist recommendations.

Upon execution of our playlist recommendation function with the specified user ID, the following playlists were recommended as most aligned with the user's historical preferences and behaviors:

- J. Period & Black Thought - The Best of The Roots
- 00-nek-en_el_cuarto_26-sp-2007
- ONLY DURAN DURAN
- Elvis 75 - Good Rockin' Tonight
- Great Dane – Alpha Dog
- 60´s 4 eva
- #facebookdown

- Led Zeppelin – How The West Was Won
- Easy Listening Blues
- Kensington – Vultures - Festival Edition

These recommendations demonstrate the effectiveness of the collaborative filtering techniques employed by our recommendation system, showcasing a diverse range of genres and artists that reflect the unique tastes and preferences of the user.

Interaction Distribution per User for Playlists Analyzing interactions per user for playlists, the distribution is highly skewed towards lower numbers, indicating that most users interact with only a handful of playlists regularly (Figure 4.6).

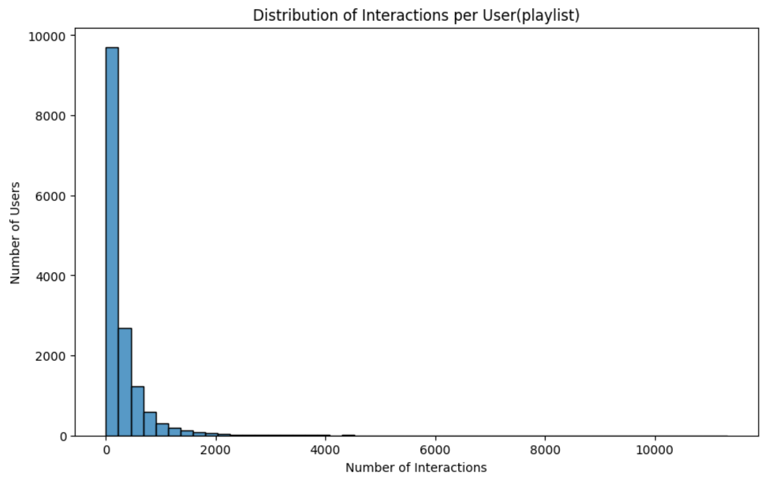


Figure 4.6: Distribution of Interactions per User for Playlists showing a high concentration of users with fewer playlist interactions.

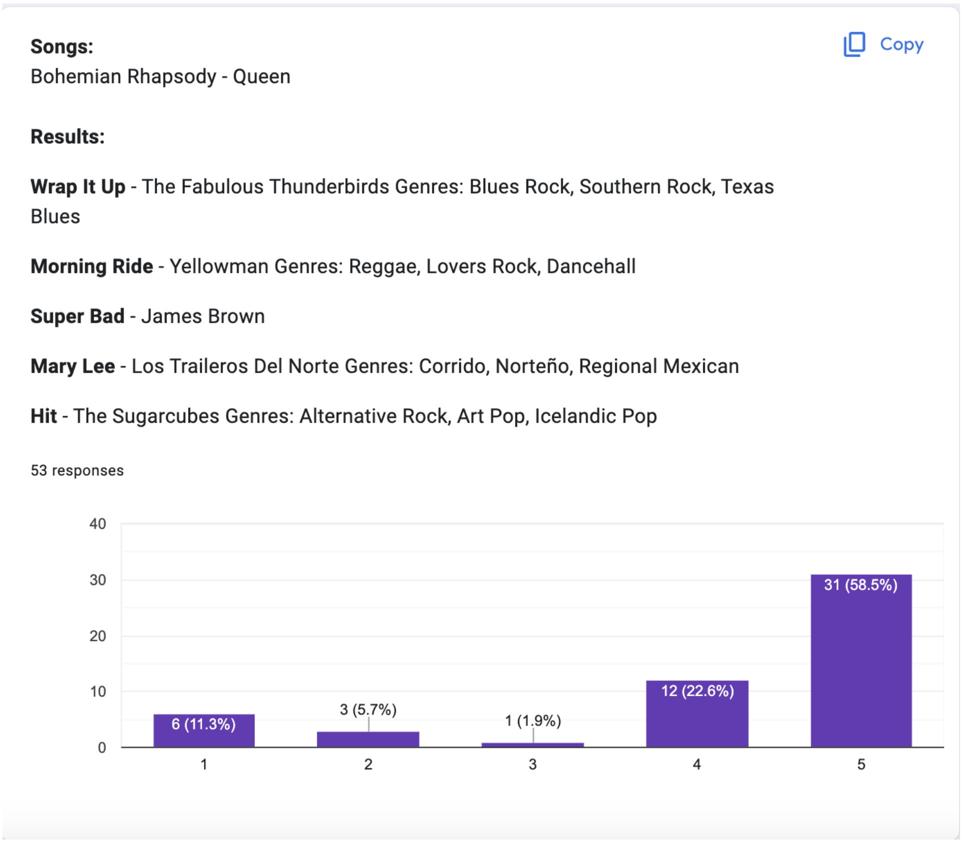
Due to the high sparsity of data and the challenge in collecting Spotify IDs, only 3 out of 63 respondents from our A/B testing questionnaire provided their IDs. Hence, we could not perform extensive testing of the playlist recommendation systems; however, the models remain robust, suggesting promising results for playlist recommendations despite these limitations.

4.2.2 Content-Collaborative Song Recommendation System

Executing the recommendation function for a specific song demonstrates the system's efficacy in creating customized song lists that accurately represent the user's individual tastes and highlight the accuracy of the hybrid model.

```python
# Example usage and output
input_song_name = '21 Guns'
recommended_songs = hybrid_recommend_songs(input_song_name,
    song_name_to_index, reduced_matrix, song_data, top_n=5)
print(f"Recommended Songs for '{input_song_name}':")
for song, artist, genres, score in recommended_songs:
    print(f"Song: {song}, Artist: {artist}, Genres: {genres},
    Score: {score:.2f}")
```

Listing 4.2: Example usage of the recommendation function and output.



Songs:
Bohemian Rhapsody - Queen

Results:

Wrap It Up - The Fabulous Thunderbirds Genres: Blues Rock, Southern Rock, Texas Blues

Morning Ride - Yellowman Genres: Reggae, Lovers Rock, Dancehall

Super Bad - James Brown

Mary Lee - Los Traileros Del Norte Genres: Corrido, Norteño, Regional Mexican

Hit - The Sugarcubes Genres: Alternative Rock, Art Pop, Icelandic Pop

53 responses

Figure 4.7: Output of song recommendations with user feedback ratings. Here, the user rated the suggestions by hearing or upon noticing.

The results emphasize the benefits of combining collaborative and content-based strategies, providing a more sophisticated method for customizing music suggestions. And the user feedback replicates it.

4.2.3 Advanced Query-Based Song Recommendation System

Upon receiving a user query, our recommendation system employs the integrated recommendation logic to produce a list of song suggestions that align closely with the user's preferences and emotional state. Below, we demonstrate the system's output for a user query requesting "death metal tracks from the 80s" with a sentiment analysis indicating a preference for intense and powerful music.

User Query:

```
"I'm looking for death metal tracks from the 80s."
```

Processed Query Output:

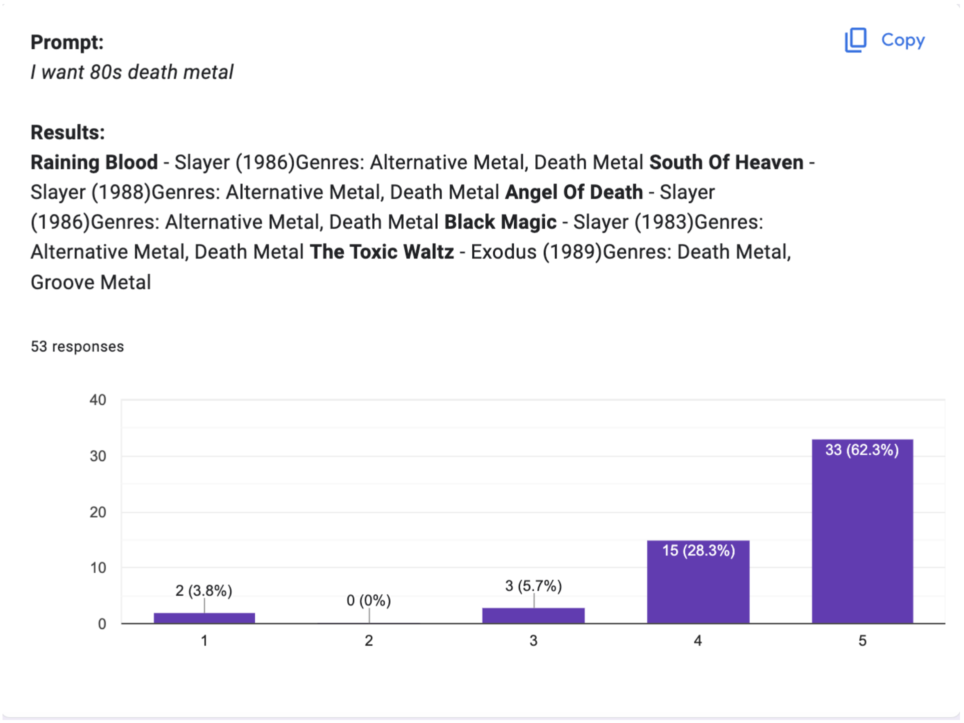
- **Genres Identified:** Death Metal
- **Moods Identified:** Intense, Powerful
- **Decades Identified:** 1980s
- **Sentiment Score:** Positive (influences the preference for high-energy songs)

Recommendations Generated: The system adjusts the scores based on the sentiment analysis and filters the songs through the identified genres, moods, and decade preferences. The top 5 recommendations, sorted by their adjusted scores, are as follows:

Track Name	Artist	Genres	Year	Adjusted Score
Raining Blood	Slayer	Alternative Metal, Death Metal	1986	1.23
The Toxic Waltz	Exodus	Death Metal, Groove Metal	1989	1.15
Black Metal	Venom	Black Metal, Black Thrash	1982	1.11
Angel Of Death	Slayer	Alternative Metal, Death Metal	1986	1.07
Elimination	Overkill	Death Metal, Groove Metal	1989	1.07

Table 4.1: Top 5 recommendations based on user query for death metal tracks from the 80s.

These results illustrate the effectiveness of our recommendation system in tailoring music suggestions that match the specified musical preferences (death metal from the 80s) and enhance the user's emotional engagement, thanks to the sentiment-driven adjustments to song popularity scores. By dynamically integrating user input, contextual data, and sentiment analysis, the system ensures that each recommendation is relevant and satisfying, enhancing the overall user experience.



Prompt:
I want 80s death metal

Results:
Raining Blood - Slayer (1986)Genres: Alternative Metal, Death Metal **South Of Heaven** - Slayer (1988)Genres: Alternative Metal, Death Metal **Angel Of Death** - Slayer (1986)Genres: Alternative Metal, Death Metal **Black Magic** - Slayer (1983)Genres: Alternative Metal, Death Metal **The Toxic Waltz** - Exodus (1989)Genres: Death Metal, Groove Metal

Figure 4.8: Output of song recommendations with query based feedback ratings. Here, the user rated the suggestions by hearing or upon noticing.

The results emphasize the benefits of combining LLM sentiment analysis with collaborative and context-based strategies, providing a more sophisticated method for customizing music suggestions. And the user 4.8 replicates it.

4.2.4 Assessment of Recommendation Systems

Methodology for User Study The participants in our user research were asked to explore music suggestions using a specially created Google form that can handle various musical tastes and input styles [**feedback_form_2024**]. This form enabled users to articulate their musical preferences using various methods:

- *General Query:* Users can provide a concise summary of their musical tastes, such as "I desire lively 80s rock music that generates a sense of joy" or "I am seeking uplifting melodic music for yoga."

- *Name of the song:* Participants can choose a specific song they like, such as "Bohemian Rhapsody by Queen," which will encourage the algorithm to search for and recommend related songs.

- *Spotify identification (Optional):* By supplying a Spotify User ID, individuals with Spotify accounts allow the system to customize recommendations according to their listening history, hence improving the customization of the suggestions.

The form used for this section of the investigation is detailed in the supplementary material, accessible through the survey link provided in the reference [34].

Feedback and System Performance The system's capacity to comprehend these diverse inputs and offer a methodical feedback procedure. Participants assessed the relevance of each music suggestion using a rating scale ranging from 1 (indicating poor relevance) to 5 (indicating great relevance) based on how well the recommendations matched their input.

The feedback from participants revealed that our recommendation system successfully provided song suggestions that closely aligned with their musical preferences. We studied the feedback for 7 particular songs for our hybrid song recsys and 8 prompts for our query-based song recsys among the 63 replies that we previously got, which were gathered using another customized form. The form may be found at this reference: [35]. The average evaluations were predominantly favorable, demonstrating the system's accuracy and the users' contentment with the recommended tunes.

The bar graph below 4.9 displays the average user ratings for each song, along with the mean, across a concise and visually informative depiction of the system's performance.

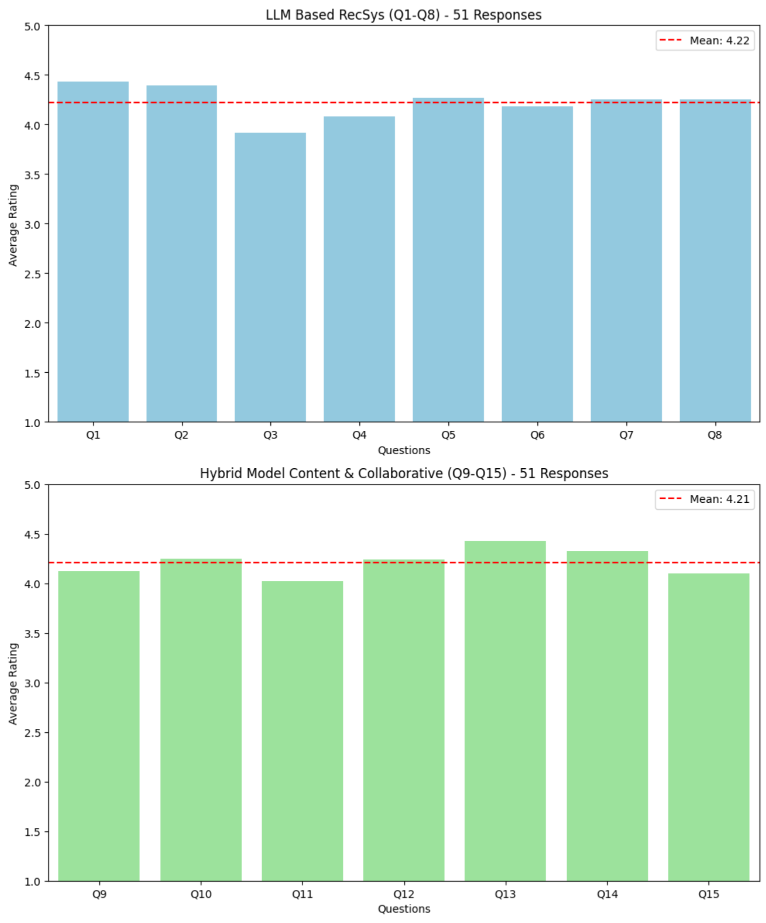


Figure 4.9: The average user ratings for song recommendations demonstrate the efficacy of the recommendation system in accurately matching varied user inputs. The mean base line indicates almost all the songs are performing near to the base line

This graph demonstrates the system's capacity to generate recommendations using diverse user inputs. It also measures the satisfaction levels for different requests, ranging from general inquiries to specific songs. These results are obtained through the structured input form mentioned as [**feedback_form_2024**].

5 Discussion and Future Works

This section presents an overview of the results obtained by combining Large Language Models (LLMs) with our query-based recommendation system and a conventional music recommendation system that uses content and collaborative filtering approaches. The main goals were to apply LLMs to improve semantic comprehension, use embeddings to concisely express musical qualities, and assess these in a hybrid recommendation system. Both the LLM-based and conventional systems obtained an average user satisfaction rating of 4.2 out of 5, demonstrating a high level of success in matching user preferences.

Although there have been positive results, there are still obstacles that prevent the system from reaching its full potential. These hurdles include a small dataset size and limitations in computational resources. The results, which are explained in detail in section 5.1, are related to specific research questions about how well LLMs and embeddings work and how they can be used in music recommendation systems. Section 5.2 outlines future work, which will focus on expanding the dataset size, enhancing computational resources, and improving system transparency and ethical standards. Such advancements are essential for refining the accuracy of the recommendations and bolstering user trust in the system.

5.1 Discussion

We have effectively implemented two advanced music recommendation algorithms for our listeners' varied musical interests. The first method employs a query-based recommendation model that leverages Large Language Models (LLMs) to perform content and semantic analysis. This technique effectively addresses the 'cold start' issue by employing sentiment analysis to customize suggestions for new users with limited initial data. The second technique utilizes content-collaborative filtering and applies embeddings to assess user interaction data. This approach effectively tackles the 'long tail' problem by proposing less popular songs that closely match user tastes.

Combined, these methods improve the user experience by guaranteeing that suggestions are thorough and tailored, including a wide range of user interests and fresh information. By integrating these models, not only are typical problems such as the long tail and cold start reduced, but the system's capacity to understand and answer complicated user inquiries with great accuracy is also enhanced.

The following segments provide a thorough discussion of the research question these systems addressed while examining their effectiveness in real-world situations.

Effectiveness of LLMs in Capturing the Semantic Meaning of Music

In response to the RQ1 question, we successfully demonstrated that our query-based LLM model can capture the semantic meaning of user queries using sentiment analysis. The model's performance looks promising, as it received an average rating of 4.2 out of 5 from 51 users. Sentiment analysis examines the emotional aspect of user queries, enabling the system to offer pertinent recommendations with less user participation and aiding in mitigating the cold start problem.

Although the query-based framework is effective, it may not fully capture the many features of musical tastes. Our data suggest that the content-collaborative filtering strategy exhibited similar performance. Integrating LLMs with content-based and collaborative filtering can potentially improve recommendation accuracy and customization. Further studies could emphasize these approaches to develop a more advanced recommendation system.

We employed the DistilBERT model from Hugging Face, a popular open-source tool, because of its optimal performance and computational economy combination. DistilBERT, a compact and highly efficient variant of BERT, is especially well-suited for applications with limited resources. Nevertheless, bigger models have the potential to provide enhanced semantic comprehension and improved accuracy in capturing a broader spectrum of musical preferences. Although these bigger models need more resources, they have the potential to offer more detailed insights into user inquiries and enhance the overall effectiveness of the recommendation system.

Our system can successfully capture the semantic meaning of user queries by overcoming these problems and utilizing diverse analytical methodologies. This connection results in more accurate and gratifying music suggestions, improving the listening experience.

The Efficacy of Embeddings in Music Recommendation Systems To address RQ2, our findings confirm that embeddings are useful for capturing latent links between songs and describing musical qualities. Embeddings greatly enhance the customization and precision of music suggestions by converting complex audio attributes and user interaction data from high-dimensional spaces into simpler low-dimensional vectors. Embeddings in the content-collaborative filtering model enable dimensionality reduction using Truncated Singular Value Decomposition (SVD), essential for effective similarity computations between songs. In addition, main Component Analysis (PCA) was used to streamline audio elements into main components, improving the system's capacity to identify and represent intricate connections within the music data. The LLM-based query model utilized embeddings to map mood and contextual information from user inquiries effectively. This enabled the system to offer tailored suggestions aligned with the user's emotional state. The use of context-aware methods in music recommendation systems has been shown to improve accuracy by Wang et al. [5], who showed that combining user interaction data and contextual metadata can make recommendations much more accurate.

Conducted on a MacBook Air M1, this study successfully handled the intricate and substantial dataset, `Data_agg` and embedding proficiently, showcasing the feasibility of running sophisticated recommendation algorithms on devices with limited resources. Despite the constraints of low computational capacity, our technique attained an average user rating of around 4.2 out of 5 for both models, as per our user surveys. This result underscores the present effectiveness of our methods but also indicates that significant enhancements might be achieved with more processing resources. Improvements in computer power would enable more advanced data processing and model training, perhaps enhancing the accuracy and customization of music suggestions even further. This scenario highlights the ability of our recommendation architecture to grow in size, and its potential to provide increasingly more accurate music selections as computational resources increase.

Hybrid System Integration of LLMs and Embeddings To address RQ3, Our study's LLM-based Query method dynamically integrates large language models (LLMs) and embeddings to optimize their respective capabilities: collecting semantic meanings and simplifying intricate data. The system employs the DistilBERT model from Hugging Face[**huggingface_distilbert**] to analyze sentiment, accurately interpreting user queries' emotional meaning. This feature allows the system to customize recommendations based on the user's mood and contextual preferences, improving the music selections' relevancy and customization.

Also, our other content-collaborative model handles and processes the `Data_agg` dataset's high-dimensional audio properties and user interactions well by embedding them. This methodology streamlines data management and enhances conventional content-based and collaborative filtering methods. These improvements enable more complex similarity calculations and provide more individualized experiences, therefore utilizing explicit and implicit preferences communicated through user interactions.

This implementation has shown that the system can provide highly tailored music suggestions even on devices with limited resources, such as the MacBook Air M1. Positive feedback for both models 4.2/5 indicates that the system generates great user satisfaction despite little computing capability. This is due to the high quality of recommendations. There is significant potential to improve system performance by increasing processing resources in the future. By adopting bigger and more advanced models in the future, the system's capacity to catch subtle variations in user preferences might be improved, resulting in even more precise and satisfying user experiences. The present utilization of this hybrid recommendation framework demonstrates a potential trajectory toward more adaptable and responsive music recommendation systems. With the increasing power of computers, we aim to incorporate both models.

Comparison of LLM-Based Query Model and Content-Collaborative Recommendation Systems For RQ4, we compared our traditional content-collaborative method with the LLM-based query method, which shows a promising improvement in user satisfaction. Based on our findings, the system's LLM-based model, incorporating the DistilBERT model for sentiment analysis, obtained an average user satisfaction rating of 4.22 out of 5 from 51 replies. This strategy marginally surpasses our other content-collaborative filtering method, which achieved a grade of 4.21 out of 5. While both algorithms performed well, the slight discrepancy emphasizes the potential benefit of integrating LLMs into the recommendation process.

Even with these positive outcomes, it's crucial to remember that the assessments were carried out on a somewhat small dataset. Although the DistilBERT model is computationally efficient, it may not comprehensively reflect the considerable complexity of user preferences on a broader scale. The current implementation requires more optimization and testing on a wider range of diverse and extensive datasets to really evaluate and improve the system's capabilities.

Ethical and Social Implications of LLMs and Embeddings in Music Recommendation Systems While addressing RQ5, we created our music recommendation algorithm, and several important ethical and societal issues were considered. Data protection was prioritized when gathering user query forms and distributing user feedback forms. We ensured that user data privacy was always protected and that consent was gained transparently by following strict data protection measures.

Transparency and Explainability: Nevertheless, the task is to improve the system's transparency further, as users sometimes enter queries that might be read in various manners or contain unclear or inappropriate material. Future system improvements will prioritize improving the processing of these inquiries to guarantee that suggestions are correct and enjoyable song recommendations for users.

Cultural Diversity: Despite the fact that our database contains a variety of music languages, computational limitations have prevented us from fully utilizing this diversity. Nevertheless, we're dedicated to enhancing the user experience, fostering cultural inclusivity, and better representing the diversity of music worldwide through expanding our system's capabilities.

Data sparsity and bias: The scarcity of user interaction data initially presented difficulties in attaining optimal suggestion accuracy. In our future upgrades, we will prioritize the integration of more comprehensive datasets to enhance the accuracy and reliability of our models. This will help minimize biases and increase the fairness of our suggestions.

5.2 Future Perspective

Currently, our methods use a query-based LLM system with semantic analysis and content-based filtering and a different system with content and collaborative filtering. Despite the promising results achieved, with the LLM-based model and content-collaborative approaches each garnering an average satisfaction rating of approximately 4.2 out of 5 in a small user survey, there remains scope for substantial improvements to refine these systems. We aim to explore the complementary advantages of combining these approaches, upon which the foundation of our current work is built.

The current LLM system, which utilizes the DistilBERT model, has demonstrated a promising capability in effectively handling semantic subtleties and emotional signals. To improve quality and individualization, future work will focus on implementing advanced linguistic models like BERT, RoBERTa, or GPT-3, widely recognized for their exceptional verbal comprehension and contextual analysis skills.

We also aim to investigate additional user features, like demographics and knowledge-based inputs, to enhance the contextual understanding of our recommendation engine. These will be integrated using richer datasets obtained via Spotify's API. This strategy will improve the significance and clarity of our recommendations and offer more recent and varied song recommendations. We aim to create a more cooperative system that utilizes improved LLMs to reveal deeper underlying meanings in user behaviors and preferences, thereby promoting more natural linkages between users and music.

Another important focus of development will be improving the efficiency of our multilingual database capabilities. We hope to ensure more cultural diversity in our suggestions by overcoming existing computational constraints and making our platform more accessible and appealing to a worldwide audience. This growth is essential for adjusting to the changing environment of global music consumption and the diverse range of users.

The anticipated improvements aim to address the constraints identified in our existing systems, including the ability to manage intricate user inputs and efficiently integrate a wide range of musical tastes. We aim to keep our system at the forefront of the music recommendation area by constantly improving our technological capabilities and features. Future recommendation systems will be able to provide even more accurate and fulfilling user experiences thanks to the combination of cutting-edge algorithms and a larger, more dynamic dataset.

6 Conclusion

This work employed a novel approach by integrating Large Language Models (LLMs) and embeddings to improve music recommendation systems. This approach effectively tackles challenges such as the cold start and long tail concerns. We used LLMs to explore the semantic nuances of user queries and embeddings to simplify intricate musical characteristics and interactions into easily understood patterns.

We introduced two hybrid models. One utilized the semantic analysis capabilities of LLMs to provide individualized content suggestions, while the other utilized content-collaborative filtering by analyzing user interactions in depth. In addition to addressing standard recommender system issues, this strategic integration greatly improved the quality and personalization of the music recommendations.

The LLM-based query model achieved a user satisfaction score of 4.22 out of 5, while the content-collaborative model scored slightly lower at 4.21, based on feedback from 51 participants. These results suggest that the LLM-based approach shows a marginal improvement over traditional methods, indicating it has the potential to enhance recommendation accuracy and personalization. The close scores also indicate the possibility that integrating LLM capabilities with traditional collaborative methods could lead to further improvements, offering a more personalized and effective way to address common issues such as the long tail and cold start problems in music recommendation systems.

The thesis identified two primary challenges that limited the maximum effectiveness of the recommendation systems: the sparsity of the datasets and the limitations of computing resources.

The incorporation of more advanced Language Learning Models (LLMs) and a more extensive investigation of embeddings of these methods will have the potential to provide a good and personalized music experience for the users. Therefore, this work establishes a basis, to motivate and encourage future advancements that will further improve the way we explore and engage with music.

Bibliography

[1] Yading Song, Simon Dixon, and Marcus Pearce. "A survey of music recommendation systems and future perspectives." In: *9th International Symposium on Computer Music Modeling and Retrieval*. Vol. 4. Citeseer. 2012, pp. 395–410.

[2] Poonam B Thorat, Rajeshwari M Goudar, and Sunita Barve. "Survey on collaborative filtering, content-based filtering and hybrid recommendation system." In: *International Journal of Computer Applications* 110.4 (2015), pp. 31–36.

[3] Laurina Zhang. "Intellectual property strategy and the long tail: Evidence from the recorded music industry." In: *Management Science* 64.1 (2018), pp. 24–42.

[4] Yan Wang et al. *Enhancing Recommender Systems with Large Language Model Reasoning Graphs*. 2024. arXiv: 2308.10835 [cs.IR].

[5] Dongjing Wang et al. "Came: Content-and context-aware music embedding for recommendation." In: *IEEE Transactions on Neural Networks and Learning Systems* 32.3 (2020), pp. 1375–1388.

[6] arXiv. "ControlRec: Bridging the Semantic Gap between Language Understanding and User Preferences in Recommendation Systems." In: *arXiv* (2023). URL: https://arxiv.org/abs/2311.16441.

[7] Wikipedia. *Spotify – Wikipedia, The Free Encyclopedia*. [Online; accessed 9-May-2024]. 2024. URL: https://en.wikipedia.org/wiki/Spotify.

[8] *Spotify for Developers*. Spotify Web API Documentation. 2024. URL: https://developer.spotify.com/documentation/web-api/.

[9] A. M. Van Der Walt. *Spotify Playlists*. Accessed: 20.03.2024. 2020. URL: https://www.kaggle.com/datasets/andrewmvd/spotify-playlists.

[10] V. Mavani. *Spotify Dataset*. Accessed: 20.03.2024. 2024. URL: https://www.kaggle.com/datasets/vatsalmavani/spotify-dataset/data.

[11] Mark Claypool et al. "Combining content-based and collaborative filters in an online newspaper." In: *Proceedings of the SIGIR Workshop on Recommender Systems*. Citeseer. 1999.

[12] Barry Smyth and Paul Cotter. "A personalised TV listings service for the digital TV age." In: *Knowledge-Based Systems* 13.2-3 (2000), pp. 53–59.

[13] Ren-ping Song et al. "A hybrid recommender algorithm based on an improved similarity method." In: *Applied Mechanics and Materials* 475 (2014), pp. 978–982.

[14] Brian Towle and Christopher Quinn. "Knowledge based recommender systems using explicit user models." In: *AAAI/IAAI*. 2000, pp. 74–77.

[15] Robin Burke. "Hybrid recommender systems: Survey and experiments." In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[16] Umut Ceylan and Ali Birturk. "Combining feature weighting and semantic similarity measure for a hybrid movie recommender system." In: *Proceedings of*

the 5th SNA-KDD Workshop on Social Network Mining and Analysis. ACM. 2011, pp. 1–9.

[17] Conor Hayes and Padraig Cunningham. "Context boosting collaborative recommendations." In: *Knowledge-Based Systems* 17.2-4 (2004), pp. 131–137.

[18] Guirong Xue et al. "Scalable collaborative filtering using cluster-based smoothing." In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2005, pp. 114–121.

[19] FS Gohari and MJ Tarokh. "Classification and comparison of the hybrid collaborative filtering systems." In: *International journal of research in industrial engineering* 6.2 (2017), pp. 129–148.

[20] Lianhuan Li, Zheng Zhang, and Shaoda Zhang. "Hybrid Algorithm Based on Content and Collaborative Filtering in Recommendation System Optimization and Simulation." In: *Scientific Programming* 2021 (2021), pp. 1–11. DOI: 10.1155/2021/7427409.

[21] Dongjing Wang et al. "Learning Music Embedding with Metadata for Context Aware Recommendation." In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ICMR '16. New York, New York, USA: Association for Computing Machinery, 2016, pp. 249–253. ISBN: 9781450343596. DOI: 10.1145/2911996.2912045. URL: https://doi.org/10.1145/2911996.2912045.

[22] Liang Xiang et al. "Temporal recommendation on graphs via long-and short-term preference fusion." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2010, pp. 723–732.

[23] Steffen Rendle et al. "BPR: Bayesian personalized ranking from implicit feedback." In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 2009, pp. 452–461.

[24] Subhash Kabbur, Xia Ning, and George Karypis. "Fism: factored item similarity models for top-n recommender systems." In: *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining* (2013), pp. 659–667.

[25] Himanshu Batra et al. "BERT-Based Sentiment Analysis: A Software Engineering Perspective." In: *DEXA 2021, LNCS 12923*. Springer, 2021, pp. 138–148. DOI: 10.1007/978-3-030-86472-9_13.

[26] Surabhi Shinde et al. "Machine Learning Based Clustering Using Spotify Audio Features." In: *2023 1st DMIHER International Conference on Artificial Intelligence in Education and Industry 4.0 (IDICAIEI)*. Vol. 1. 2023, pp. 1–5. DOI: 10.1109/IDICAIEI58380.2023.10406332.

[27] Wenqi Fan et al. "Recommender systems in the era of large language models (llms)." In: *arXiv preprint arXiv:2307.02046* (2023).

[28] Abu Mohammad Taief. *Music_Final*. https://github.com/MaidenTaief/Music_Final. Accessed: 2024-05-15. 2024.

[29] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[30] Peter J. Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.

[31] T. Caliński and J. Harabasz. "A dendrite method for cluster analysis." In: *Communications in Statistics-theory and Methods* 3.1 (1974), pp. 1–27.

[32] David L. Davies and Donald W. Bouldin. “A cluster separation measure.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1979), pp. 224–227.

[33] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” In: *arXiv preprint arXiv:1910.01108* (2019). URL: https://arxiv.org/abs/1910.01108.

[34] Google Forms. *User Feedback Survey on Music Recommendation System*. https://docs.google.com/forms/d/1PWLW23TFRQlUGMZFWJwrwFU9KsVWS066CluyqtfU6k/viewform. Accessed: 2024-05-13. 2024.

[35] Google Forms. *Evaluation Survey for Music Recommendation System*. https://docs.google.com/forms/d/1ZnZcr8E_zwMzv5j1k-31iBm8ok5XnO3WYsWI73HeXyc/viewform. Accessed: 2024-05-13. 2024.

Appendix

.1 Thesis Task Description

UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT - The Arctic University of Norway

Application of LLMs and Embeddings in Music Recommendation Systems

Abu Mohammad Taief

Thesis for Master of Science in Technology / Sivilingeniør

Background

Recommender systems play a crucial role in the music industry. They enable users to discover new music that aligns with their preferences. These systems predict music preferences using user behavior, musical features, and metadata. Traditional recommender systems often rely on collaborative filtering or content-based filtering approaches. Collaborative filtering makes recommendations based on the listening patterns of similar users, creating a sense of community taste. Conversely, content-based filtering focuses on the properties of the music itself, such as genre, tempo, and instrumentation, to make suggestions.

However, these approaches have limitations in capturing the semantic meaning of music and providing personalized recommendations. Collaborative filtering can suffer from the cold start problem. This problem arises when new songs or users with unique tastes cannot match existing data. Additionally, it struggles to recommend less popular tracks, which fall under the "long-tail distribution" of music. On the other hand, content-based filtering can be excessively simplistic. It fails to capture the complex and subjective qualities that give music its semantic meaning and emotional resonance.

Problem description

Traditional recommender systems in the music industry have limitations. These models should be capable of understanding and predicting user preferences more accurately. The semantic meaning of music, a combination of lyrics, composition, and the emotions they evoke, needs to be adequately captured by existing methods. Personalization also remains a challenge, as users have diverse and evolving musical tastes that are not easily categorized.

This thesis proposes exploring the use of large language models (LLMs) and embeddings in the context of music recommendation tasks to address existing issues. LLMs have proven their capabilities in understanding and generating human-like text, which may help decipher the semantic content of lyrics and user reviews. Embeddings, which refer to vector representations of music items in a high-dimensional space, can capture the subtle relationships between different pieces of music and user preferences.

Scope and limitations

- Focus on the academic part and research problems.
- Implementing a commercial product is out of scope.
- Proof-of-concept implementation of algorithms are expected to emerge.

Research questions

- To what extent can LLMs effectively capture the semantic meaning of music?
- How can embeddings effectively represent musical features and capture latent relationships between songs?
- How can a hybrid recommendation system that integrates LLMs and embeddings be effectively designed and developed?
- How does the proposed hybrid recommendation system compare to traditional approaches regarding accuracy, personalization, and user satisfaction?
- What are the ethical and social implications of integrating LLMs and embeddings in music recommendation systems?

Objectives

- Review the current state of research on music recommendation systems and the limitations of traditional approaches.
- Explore the potential of LLMs in understanding the semantic meaning of music and generating personalized recommendations.
- Investigate using embeddings to represent musical features and capture latent relationships between songs.
- Develop and evaluate a hybrid recommendation system that integrates LLMs and embeddings.
- Analyze the proposed system's performance in terms of accuracy, personalization, and user satisfaction.

Dates

Date of distributing the task:	<08.01.2024>
Date for submission (deadline):	<21.05.2024>

Contact information

Candidate	Abu Mohammad Taief ata059@post.uit.no
Supervisors at UiT-IDBI	Bernt Arild Bremdal bernt.a.bremdal@uit.no
	Shayan Dadman shayan.dadman@uit.no
	Kalyan R Ayyalasomayajula kalyan.r.ayyalasomayajula@uit.no

General information

This master thesis should include:

❉ Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, datasets, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis

❉ Selection & investigation of actual materials

❉ Development (creating a model or model concept)

❉ Experimental work (planned in the preliminary work/literature study part)

❉ Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 3 weeks. It should include bullet points 1 and 2 in "The work shall include", and a plan of the progress. The preliminary study may be submitted as a separate report or "natural" incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis, emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgments, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software / source code or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

The candidate is required to demonstrate continuous progress in developing and completing the work. Continuous progress is defined as consistent, substantive advancement in research, analysis, writing, and other activities directly related to fulfilling the thesis requirements determined by the supervisory committee.

If the candidate fails to demonstrate continuous progress, as evidenced by regularly requesting help without showing substantive independent effort or failing to engage with the supervisors or the thesis work, the supervisors may limit or withdraw support and assistance.

Submission requirements

This thesis should result in a final report with an electronic copy of the report including appendices and necessary software, source code, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered according to the current faculty regulation. If there is an external company that needs a copy of the thesis, the candidate must arrange for it. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The supervisor(s) should receive a copy of the the thesis prior to submission of the final report. The final report with its appendices should be submitted no later than the decided final date.