Jurnal Elektronika dan Telekomunikasi (JET), Vol. 25, No. 1, August 2025, pp. 20-27
Accredited by RISTEKDIKTI, Decree No: **158/E/KPT/2021**
doi: 10.55981/jet.660

Implementation of Bidirectional Encoder Representations from Transformers in a Content-based Music Recommendation System for Digital Music Platform Users

Fadil Abdillah Suyudi[a], Muhammad Ariful Furqon[b,*], Qurrota A'yuni Ar-ruhimat[b]

[a]Department of Information Technology
Universitas Jember
Jalan Kalimantan No. 37, Sumbersari
Jember, Indonesia
[b]Department of Informatics
Universitas Jember
Jalan Kalimantan No. 37, Sumbersari
Jember, Indonesia

Abstract

Digital music platform users can access millions of songs from various genres and artists through music streaming services. However, with so many music platforms available, users often need help finding songs that suit their preferences. This study presents a music recommendation system that utilizes lyrical analysis to provide users with relevant song suggestions based on selected lyrics. The system employs a two-pronged approach: the Term Frequency-Inverse Document Frequency (TF-IDF) method for initial feature extraction and the IndoBERT model for advanced contextual representation of song lyrics. A dataset of 8,944 Indonesian language songs was compiled using scraping techniques from various sources. The recommendation process is driven by cosine similarity calculations between the lyrics of the selected songs and the entire dataset, enabling the identification of songs with similar themes and messages. Model evaluation through a five-fold Multi-Class Cross-Validation (MCCV) approach yielded promising results, indicating high precision, recall, and F1 scores. The study results show that the system built can provide recommendations with good precision performance with Precision@k values varying between 0.7965 to 0.8371, Recall@k values ranging from 0.8017 to 0.8204, and F1-score@k values varying between 0.8083 up to 0.8190. Overall, the model shows strength in providing accurate recommendations and good performance stability

Keywords: Content-based Music Recommendation, Cosine Similarity, BERT, Monte Carlo Cross-Validation

I. INTRODUCTION

In recent years, digital music platforms have revolutionized how people consume music, offering users instant access to millions of songs across various genres and artists [1] . Services like Spotify, YouTube Music, and Apple Music have grown exponentially, with advanced recommendation algorithms to enhance the user experience [2] . However, with overwhelming content, users often need help discovering music that aligns with their preferences. Traditional recommender systems, primarily based on user behavior and preferences, sometimes fail to capture the full complexity of a listener's needs in different scenarios [3] . Some contextual information has been proven to help recommender systems better understand and satisfy users in real time [4].

Music recommendation systems are designed to help users discover a diverse array of songs, enhancing the personalization of their listening experience [5] . Several core methods, such as collaborative, content-based, and hybrid filtering, have been widely applied to develop these systems [6]. Researchers have introduced advanced approaches to improve recommendation accuracy and relevance as the field evolves. One of the most promising developments is using machine learning models to interpret and process text data more effectively [7] . Among these models, Bidirectional Encoder Representations from Transformers (BERT), developed by Google, has achieved state-of-the-art results in numerous natural language processing (NLP) tasks [8]. Given its ability to capture nuanced meaning from text, BERT presents a compelling opportunity to enhance music recommendation systems further, making them more context-aware and responsive to users' preferences [9].

Several studies have highlighted the limitations of traditional recommendation methods in capturing user preferences dynamically. Abdi et al. presented a survey on collaborative filtering using matrix factorization, which improved recommendation accuracy by breaking the data into latent factors [10]. While this method has been effective, it often needs help with cold-start problems or when insufficient user interaction data is available [11]. Deldjoo et al. emphasized that traditional content-based methods, which rely on manually designed content features, often fail to capture deeper user preferences [12] . To address these limitations, researchers have turned to machine learning techniques,

* Corresponding Author.
Email: ariful.furqon@unej.ac.id
Received: July 24, 2024 ; Revised: September 18, 2025
Accepted: December 18, 2024 ; Published: August 31, 2025

which provide a more flexible and automated approach for analyzing and predicting user preferences [13]–[15].

In particular, integrating NLP models into recommender systems has garnered increasing attention. Elkahky et al. explored the use of deep learning models for extracting richer features from text data, such as song lyrics and reviews, improving the relevance of recommendations [16]. Similarly, Zhu et al. demonstrated the potential of BERT to fine-tune domain-specific tasks, achieving notable success in capturing complex contextual relationships [17]. These advancements suggest that leveraging text-based data through NLP models, especially BERT, could significantly enhance the performance of content-based recommendation systems.

This study proposes a method that leverages the vector representations of text generated by the BERT model to calculate content similarity. By analyzing the semantic relationships within the text, this method can recommend content with the highest similarity to user preferences. This approach addresses the limitations of traditional keyword-based methods, which often need help to capture deeper contextual meaning [11]. The results of this study are expected to advance the field of recommendation systems significantly, particularly in improving the precision and personalization of content-based recommendations.

II. METHODS

This study consists of several stages: data collection, preprocessing, feature extraction, BERT model implementation, system development, and testing.

A. Data Collection

The data collection stage was conducted using the scraping technique, an automated method for gathering data from various online sources to facilitate research analysis [18]. In this study, music data was collected utilizing the Spotify API to obtain song metadata, such as artist names, album information, and track popularity. Additionally, Genius lyrics were leveraged to extract song lyrics, providing rich textual content for subsequent analysis. This dual approach ensured a comprehensive dataset, incorporating both quantitative and qualitative information to enhance the overall quality of the research [19].

B. Data Preprocessing

Data preprocessing involved several crucial steps, such as cleaning and preparing the data before it can be utilized in the model [20]. Data cleaning was performed to remove any irrelevant, duplicate, or incomplete records, enhancing the overall dataset quality [21]. This was followed by normalization, standardizing data formats such as converting text to lowercase and ensuring consistent punctuation and spacing [22]. Tokenization was then applied to split the text data into individual tokens or words, facilitating more accessible analysis in subsequent stages [22].

To further refine the dataset, stop word removal eliminated common words that did not add significant meaning, allowing the focus to remain on more informative terms [23]. Finally, stemming or lemmatization reduced words to their root forms, ensuring that word variations were treated as the same term [24]. The dataset was effectively transformed into a format suitable for model training and analysis by executing the above preprocessing steps [25].

C. Feature Extraction

After data preprocessing, text features were extracted using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This approach quantifies the importance of a word within a document relative to the entire corpus [26]. This study employs TF-IDF to identify significant words in song lyrics, ensuring that frequently used yet uninformative words are down-weighted [27]. The TF-IDF values were computed for each word in the corpus, producing a vector representation of the text [26].

D. BERT Model Implementation

The BERT model utilized in this study was IndoBERT, a variant of BERT specifically trained for the Indonesian language [28]. The implementation of IndoBERT follows a series of steps described in Figure 1. The process started with applying the model to generate deep contextual embeddings for song lyrics. These embeddings capture the semantic meaning of the lyrics in a way that goes beyond simple word matching.

This study generates music recommendations by combining the TF-IDF similarity values with the BERT-based similarity values. This hybrid approach ensures that surface-level and deeper contextual meanings are considered when measuring song similarity [17]. In a music recommendation system, it is crucial to assess how similar the lyrics of one song are to another to provide users with relevant recommendations [29]. By leveraging the strengths of both methods, this approach aims to deliver more accurate and meaningful music suggestions based on user preferences [17].

E. Model Evaluation

The model evaluation aims to assess the performance of the lyrics-based music recommendation model using BERT. The Monte Carlo Cross-Validation (MCCV) method was employed for evaluation to ensure robust results, which helps to avoid overfitting and promotes good model generalization [30]. MCCV involves repeatedly splitting the dataset into random training and testing subsets. In each iteration, the dataset was divided, typically in an 80-20 ratio, with the model trained on the training set and evaluated on the test set. This process was repeated 100 times, and the performance metrics Precision@k, Recall@k, and F1-score@k—were calculated for each split. Precision@k,

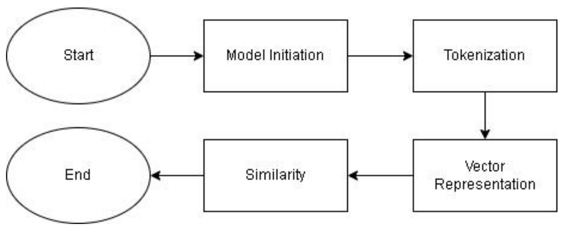

Figure 1. BERT implementation.

Recall@k, and F1-score@k are defined in (1), (2), and (3), respectively.

$$Precision@k = \frac{N\ relevant\ items\ in\ top - k\ reccomendation}{k} \quad (1)$$

$$Recall@k = \frac{N\ relevant\ items\ in\ top - k\ reccomendation}{Total\ number\ of\ relevant\ items} \quad (2)$$

$$F1 - score@k = 2 \cdot \frac{Precision@k \cdot Recall@k}{Precision@k + Recall@k} \quad (3)$$

Precision@k measures the proportion of relevant items among the top-k recommended items, providing insight into the accuracy of the recommendations. Recall@k captures the proportion of relevant items retrieved from all available relevant items, emphasizing completeness. The F1-score@k, calculated as the harmonic mean of Precision@k and Recall@k, balances these two metrics and comprehensively evaluates the model's performance. The results from all MCCV iterations were aggregated, and each metric's mean and standard deviation were reported to ensure reliable and robust performance estimates.

F. System Development

The system development was carried out using Streamlit Python, leveraging the results of the similarity calculations to generate music recommendations. The process begins by retrieving the album cover art for the songs, based on their titles in the dataset, using the Spotipy library, which interacts with the Spotify API. A function was then created to display the top 10 song recommendations based on the highest similarity values. This user-friendly interface allows users to visually explore and interact with the recommended songs, enhancing the overall recommendation experience.

III. RESULT AND DISCUSSION

A. Data Collection

The dataset for this study was collected using scraping techniques, which involved extracting data from relevant sources. Python libraries, specifically Spotipy and lyricsgenius, were utilized to gather the necessary song information. Spotipy retrieved song metadata from Spotify, including the song title, artist, and track ID. Meanwhile, lyricsgenius was used to obtain the corresponding song lyrics from the Genius platform.

The final dataset consists of 8,944 Indonesian songs, each containing attributes such as the artist's name, title, and lyrics. These attributes are crucial as the artist's name and song title were the input for users to request song recommendations. The extraction process for Spotify playlists began by collecting information about the songs in the playlists, such as the title and artist, before saving this information in a JSON file. This structured data was then utilized in the recommendation system to provide personalized song suggestions. Table 1 presents the results of the dataset scraping process and their impact on the recommendation system's performance.

TABLE 1
DATASET

Id	title	artist	lyrics
0	A N G	Naff	Seluruh hati tlah kudatangi\nHan...
1	A Ya Ya	AB Thre	Ku tak ingin kau katakan cinta\nKu tak ingin k..
2	A Ye O (Tamasya)	Soul ID	Lalalalaa heeeey… Lalalalaa heeeey\nHari ini a...
3	A-A-A-A-A	Naff	kamu datang di saat tepat\nbebaskan jiwa yang ...
4	A...A...A...	Rif	Kau... tak tahu yang terjadi\nterkunci jauh..
...	….	…	……
8943	98	Netral	1998 tahun gila tahunnya macan\nPamer taring ...

B. Data Preprocessing

The data preprocessing stage is crucial for cleaning, preparing, and transforming the text to ensure it aligns with the required format for text analysis. This process involves removing unnecessary characters, standardizing formats, and applying tokenization, lowercasing, and punctuation removal techniques [26]. The results of the preprocessing step are summarized in Table 2, highlighting the changes made to the text.

C. Feature Extraction

After cleaning the dataset, the next step is feature extraction, which aims to identify and extract significant features from the song lyrics that can be utilized for further analysis or modeling. In this study, the TfidfVectorizer library was employed for this task.

The process began by initializing the cleaned lyric text and transforming it into a TF-IDF vector representation. This representation calculated the importance of each word within the document relative to the entire corpus [26] . The result was a TF-IDF matrix, as shown in Table 3, which stored the TF-IDF weights for each term across the documents. Below is a sample of the matrix built from the song "*Abadikah Tragedi*" lyrics.

TABLE 2
PREPROCESSING RESULTS

Before	After
Sudah tujuh samudera ku arungi bersama dengan dirimu\nMerangkai hidup dengan suka duka\nSangat tak bisa ku pahami dan mengerti yang terjadi kini\nKau menjauh dan pergi tinggalkanku\n	sudah tujuh samudera ku arungi bersama dengan dirimu merangkai hidup dengan suka duka sangat tak bisa ku pahami dan mengerti yang terjadi kini kau menjauh dan pergi tinggalkanku

TABLE 3
TF-IDF MATRIX

Term	Weight
tragedi	0.693512
aku	0.246685
bakar	0.194413
terhumban	0.192798
biar	0.181583
begini	0.177662
suci	0.176397
selubung	0.148931
…	….
rasa	0.044870

From the results of the TF-IDF calculations, certain words stand out due to their high weights, offering insight into the key themes and messages conveyed in the song. The word "tragedi" has the highest weight, 0.693512, indicating that the theme of tragedy is central and significant in the song's lyrics, likely pointing to a dramatic or emotional event. The word "aku," weighing 0.246685, suggests a personal perspective commonly used in songs to express self-reflection or the songwriter's experiences.

The TF-IDF analysis helps uncover keywords that define the theme, tone, and main message of a song's lyrics, highlighting unique and meaningful elements in the song's context [31] . After obtaining the TF-IDF vector representation of the documents, the next step involved calculating the similarity value between the documents based on the TF-IDF matrix. Document similarity measures how closely two documents are related to each other. One common metric is cosine similarity [32], [33] . Cosine similarity measures the cosine of the angle between two vectors in the TF-IDF vector space, producing a value between 0 and 1. A value of 1 indicates that the documents are very similar, while 0 suggests they are very different. This was calculated using the dot product of the two vectors divided by the product of their magnitudes [33].

D. IndoBERT Implementation

The IndoBERT model, a BERT variant trained explicitly for the Indonesian language [34] , was implemented in this study to provide contextualized representations of song lyrics. The process began by initializing and loading the IndoBERT model using the Hugging Face Transformers library [35] . The preprocessed song lyrics were then tokenized, converting the text into a format that the IndoBERT model can process. This tokenization step transformed the lyrics into vector representations that capture the context of words within the entire text.

After tokenization, a vector representation of the lyrics was obtained, creating a feature profile for each song. These feature profiles are then compared using the cosine similarity metric to identify songs with similar lyrics [17]. When a user selects the lyrics of a song, the recommendation system leverages this similarity to suggest songs with closely matching lyrical content, thereby improving recommendation relevance and user satisfaction.

The BERT embedding is shown in Figure 2, which starts by feeding the text into the BERT tokenizer, which transforms the input text into a PyTorch tensor. The tokenizer truncates any text exceeding the length of 512 tokens and pads shorter texts to ensure uniform input length [36] . Once tokenized, the input tensor is passed through the IndoBERT model, generating a vector representation (embedding) for each token in the text. The embeddings from the last hidden state of the model are averaged to create a single embedding vector for each song lyric. This embedding is then stored in a NumPy array for further analysis.

BERT embeddings were generated and repeated for every lyric in the dataset by applying the get_bert_embedding function to each entry in the song lyrics column. These embeddings, stored in a NumPy array, formed the basis for the subsequent similarity calculation and served as the core input for building the recommendation system.

Algorithm 1 Bert Embedding

```
Input:
    A song lyric $x = [w_1, w_2, \dots, w_n]$, where $w_i$
    represents each word in the lyric.

Process:
    # Tokenization
    $tx = T(x)$ where $t_x = [t_1, t_2, \dots, t_k]$

    # $k$ is the number of tokens after
    truncation and padding (with $k \le 512$).

    # BERT Embedding
    $H_x = M(t_x)$ where $H_x = [h_1, h_2, \dots, h_k]$
    and $h_i \in \mathbb{R}^d$

    # Mean Pooling to Get Song Embedding
    $v_x = \frac{1}{k}\sum_{i=1}^{k} h_i$ where $V_x \in \mathbb{R}^d$

    # Embedding Matrix for All Lyrics
    $V = \begin{bmatrix} V_{x1} \\ V_{x2} \\ \vdots \\ V_{xn} \end{bmatrix}$ where $V \in \mathbb{R}^{n \times d}$

Output:
    Embedding matrix $V \in \mathbb{R}^{n \times d}$, where each row
    corresponds to embedding a song lyric.
```

Figure 2. BERT Embedding

Once the BERT embeddings for each song lyric have been generated, the next step was to compute the similarity values between the embeddings. The cosine similarity method measures how close the vector representations are to each other [33] . BERT embeddings contain rich contextual information, making the similarity measure more accurate in capturing the meaning and theme of the lyrics. After generating the BERT embedding vectors, each vector was compared to others using cosine similarity, producing a similarity matrix. This matrix captured how similar each song lyric is to others in the dataset. The similarity values were calculated using the dot product of the vectors, normalized by their magnitudes [33].

The combination of TF-IDF and BERT methods enhances the accuracy of the recommendation [17]. TF-IDF captures word frequency and highlights important terms, while BERT provides a deep contextual understanding of the lyrics [37] . The system provides more robust and relevant song recommendations by averaging the similarity values from both methods [36]. The recommendation process started by searching for the song index based on the song title. Once identified, the function combined the similarity scores from both the TF-IDF and BERT methods, averaging them to produce a final similarity score for each pair of songs. This allows the system to balance both frequency-based and context-based measures of similarity [36].

Finally, the top 10 songs with the highest combined similarity values were identified, excluding the original.

Table 4. presents the recommendations based on the song "*Abadikah Tragedi*".

E. Model Evaluation

The model evaluation began with a descriptive analysis of the similarity values obtained from the recommendation system. This analysis is crucial for understanding how similar or different the song lyrics are, which, in turn, informs the recommendation process. Three main statistical metrics were used: the mean, median, and standard deviation. The descriptive analysis helps identify general trends, such as how closely related the songs are on average, and detect any outliers [38] —songs with unusually high or low similarity values. This analysis is also fundamental for determining a threshold value that separates relevant from irrelevant recommendations. A carefully chosen threshold allows the system to decide which songs to recommend to the user based on how similar their lyrics are to the song the user has selected. Songs that meet or exceed the threshold are considered relevant for recommendation, while those that fall below are excluded. A similarity distribution descriptive analysis was conducted, and the results are shown in Figure 3.

Based on Figure 3, it is evident that the distribution of similarity values within the dataset follows a pattern where it increases steadily up to a similarity value of 0.6. This indicates that a significant portion of the items in the dataset exhibit a relatively high degree of similarity. However, as the similarity values surpass 0.6 and approach 0.7, the graph shows a decline, suggesting that the number of items with high similarity decreases beyond this point. This reflects a shift in the distribution, where fewer items maintain such a high level of similarity, leading to more variation or dissimilarity among items beyond this threshold.

TABLE 4
RECOMMENDATION RESULTS

Title	Artist	Similarity
Hentian Ini	XPDC	0.7102
Biar Putih Tulang	Dinamk	0.7068
Ikuti	Dan	0.7039
Kekal	Spider	0.7033
Ratib Orang Pinggiran	Saleem	0.7027
Kembalikan Indah	Siti Nuhaliza	0.7023
Cinta Kenangan Silam	XPDC	0.6997
Malam dan Semalam	Slam	0.6995
Tak Esok Lusa	Spider	0.6987
Setelah Aku Kau Miliki	Shima	0.6978

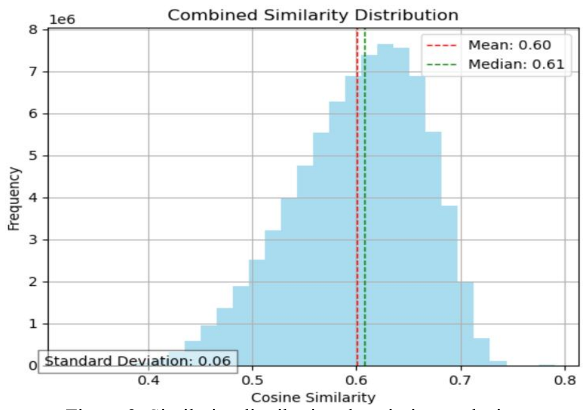

Figure 3. Similarity distribution descriptive analysis.

The statistical analysis further clarifies the distribution of similarity values. The mean similarity value of 0.6 indicates that, on average, the items in the dataset share a moderate level of similarity. This suggests that there is generally a significant overlap between the themes or content of different songs for a lyrics-based music recommendation system. Meanwhile, the median similarity value of 0.61 is slightly higher than the mean, suggesting a positive skewness in the distribution. This means that while most items are around the average similarity, some items have higher-than-average similarity, pulling the median upwards.

Furthermore, a standard deviation of 0.06 is relatively small, showing that the similarity values do not deviate far from the mean, indicating consistency in the level of similarity across most items. The limited variation suggests that despite some diversity in the dataset, most songs share a fairly consistent degree of similarity with others, which is beneficial for a recommendation system as it ensures that recommendations are consistent and consistent.

The MCCV method was employed to assess the performance of the lyric-based music recommendation model [30]. By splitting the dataset into multiple folds (or subsets), MCCV enables the model to be tested across different data sections to avoid overfitting and ensure generalization [39]. In this study, the dataset was divided into 5 folds, where each fold was used as the testing set, whereas the rest were used for training. This iterative process allows for a thorough evaluation of the model's performance across all subsets of data.

In each MCCV iteration, three key evaluation metrics i.e., Precision@k, Recall@k, and F1-score@k, were calculated to assess the accuracy and quality of the recommendations [30]. These metrics evaluate how well the model identifies relevant songs for users, particularly focusing on the top 4,472 recommendations (as represented by the variable $k = 4{,}472$). For each user in the test set, the actual liked items were compared with the predicted recommendations based on similarity. The list of recommendations was sorted by descending similarity values, ensuring that the most similar items are prioritized.

The results of Precision@K, Recall@K, and F1-score@K for each fold were then averaged to comprehensively evaluate the model's performance in each MCCV iteration. These metrics provide insights into the model's ability to balance precision (the relevancy of recommended songs) with recall (the model's ability to recommend all relevant songs) while maintaining a good overall performance [30] . This multi-fold validation process ensures that the model is robust and capable of delivering reliable recommendations across different subsets of data. The results of the MCCV evaluation provide a solid benchmark for comparing and improving the recommendation system's performance, which is presented in Table 5.

Based on the evaluation results of the five MCCV iterations listed in the table, the performance of the similarity-based recommendation model can be assessed. Each iteration shows that the Precision@k value varies between 0.7965 and 0.8371, with the highest value in

TABLE 5
MCCV EVALUATION RESULTS

Iteration	Precision@k	Recall@k	F1-score@k
1	0.8371	0.8017	0.8190
2	0.8096	0.8093	0.8095
3	0.8195	0.8048	0.8121
4	0.7965	0.8204	0.8083
5	0.8242	0.8022	0.8131
Average	0.81738	0.80768	0.8125

the first iteration. This indicates that the model can identify relevant items fairly well. The Recall@k value varies between 0.8017 and 0.8204, with the highest value in the fourth iteration. This shows that the model is consistently able to detect relevant instances. It suggests that while the recommended items were relevant, the model might be overlooking numerous other relevant items that could also be suggested.. F1-score@K, which is the harmonic average of Precision@K and Recall@K, shows a moderate value ranging from 0.8083 to 0.8190. The first iteration achieved the highest value, indicating a good balance between accuracy and recall.

To evaluate the effectiveness of the proposed approach, we conducted a benchmarking analysis comparing our model with various traditional and deep learning models from prior studies. Metrics such as Precision, Recall, and F1-score were used to assess the performance of each model, with higher values indicating better alignment between predictions and actual results. The proposed model, utilizing BERT with TF-IDF features, was compared against models like Naïve Bayes, Random Forest, XGBoost, and CNN, each implemented with different feature extraction methods (Count Vectors and TF-IDF). The detailed results of this comparison are presented in Table 6.

The comparison results demonstrate that the model developed in this study, which combines BERT with TF-IDF features, significantly outperforms all other models across the evaluated metrics (Precision@k, Recall@k, and F1-score@k), achieving consistent scores of 0.81–0.82. This highlights the strength of transformer-based contextual embeddings in capturing semantic nuances, further enhanced by the effective feature representation provided by TF-IDF. In contrast, traditional machine learning models like Naïve Bayes and Random Forest perform better with Count Vectors than with TF-IDF, suggesting that these simpler models are more effective with basic frequency-based features. Among these, Naïve Bayes with Count Vectors achieves the highest performance (0.72), followed closely by Random Forest (0.70).

TABLE 6
MODEL COMPARISON

Model	Precision	Recall	F1-score
This study (BERT,TF-IDF)	**0.82**	**0.81**	**0.81**
Naïve Bayes, Count Vectors [40]	0.72	0.72	0.72
Naïve Bayes, TF-IDF [40]	0.67	0.67	0.67
Random Forest, Count Vectors [40]	0.70	0.70	0.70
Random Forest, TF-IDF [40]	0.67	0.67	0.67
Xgboost, Count Vectors [40]	0.65	0.66	0.66
Xgboost, TF-IDF [40]	0.63	0.63	0.63
CNN, Count Vectors [41]	0.65	0.69	0.67

Interestingly, XGBoost shows the lowest performance regardless of feature type, with scores ranging from 0.63 to 0.66. This indicates that XGBoost may not be well-suited for this task or dataset. On the other hand, CNN, a deep learning model leveraging Count Vectors, shows moderate results (F1-score@k = 0.67), suggesting that it benefits from sequential patterns but falls short compared to BERT. Overall, the results emphasize the superiority of transformer-based models over traditional and other deep learning approaches, particularly when combined with robust feature extraction techniques.

Based on BERT evaluation, it clearly shows that the recommendation model reliably delivers song suggestions that are both accurate and relevant to the users' preferences.The stability of performance metrics across iterations reflects the model's robustness and reliability. Despite slight variations, there is no significant drop in performance between iterations, indicating the system's consistency during testing. While high precision is a strength, enhancing recall can lead to even more comprehensive recommendations.

F. System Development

The music recommendation system was designed and implemented with a user-friendly interface in the System Development phase. The development process started with Spotipy, a Python library that interacts with the Spotify Web API to retrieve album covers for each song title in the dataset [42] . The next step was to develop a function to display the top 10 song recommendations based on the similarity values calculated using the combined TF-IDF and BERT models. The interface of the developed music recommendation system is visualized in Figure 4.

The result of this development is visualized in Figure 4, which presents the user interface built with Streamlit, a popular Python library for creating web applications. The interface is simple yet effective, allowing users to interact with the system effortlessly. At the top of the interface, there is a dropdown menu where users can select or type the title of a song from the dataset. Once a song is selected—such as the song "*Abadikah Tragedi*" in the example—the user can click

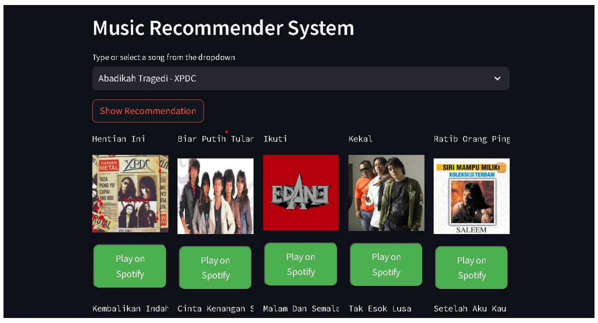

Figure 4. Interface of recommendation system.

the "Show Recommendation" button, which is styled in red, to view the recommendations.

When the button is clicked, the system generates a list of song recommendations based on their similarity to the selected song. These recommendations are displayed neatly, with each recommended song occupying a column. At the top of each column is the song title, followed by the album cover image in the middle, enhancing the interface's visual appeal. Below the album cover is the "Play on Spotify" button, which users can click to open and listen to the recommended song on Spotify directly. This clean, structured interface provides a seamless experience for users, making it easy to discover new music similar to their preferences. Combining an intuitive design and functional integration with Spotify helps the system deliver personalized and relevant song recommendations in a visually engaging format.

IV. Conclusion

The research results show that developing a music recommendation system using a combination of TF-IDF and BERT successfully provides recommendations to digital music platform users. The system can produce the top 10 recommendations with the highest level of similarity, which can help users find songs that match their preferences. The evaluation results show that the model shows consistent performance. The Precision@k value varies between 0.7965 and 0.8371, indicating that the model can identify relevant items. The Recall@k value ranges from 0.8017 to 0.8204, indicating that the model is consistently able to detect relevant items, although it is possible that some relevant items were missed. The F1-score@k values varied between 0.8083 and 0.8190, reflecting a good balance between accuracy and recall. Overall, the model shows robustness in providing accurate recommendations and good performance stability across iterations without significant performance degradation, indicating consistency in testing.

Future works enhancing the music recommendation system can focus on several key areas. Integrating user-based collaborative filtering would personalize recommendations based on user preferences and listening history. Expanding support for multilingual lyrics and implementing context-aware recommendations based on factors such as mood or time of day could broaden the system's usability and relevance. Improving model accuracy through advanced architectures like RoBERTa or XLNet and hyperparameter tuning could lead to better precision and recall in recommendations.

Declarations

Conflict of Interest

The authors have declared that no competing interests exist.

CRediT Authorship Contribution

Fadil Abdillah Suyudi: Conceptualization, Investigation, Software, Validation, Writing - Original Draft; Muhammad 'Ariful Furqon: Resources, Methodology, Writing - Review & Editing, Supervision, Funding acquisition; Qurrota A'yuni Ar-ruhimat: Formal analysis, Data Curation, Supervision.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

Acknowledgment

The authors would like to express their sincere gratitude to the Institute for Research and Community Service (LP2M) at Universitas Jember (UNEJ).

References

[1] P. Wikström, *The music industry: Music in the cloud*. John Wiley & Sons, 2020.

[2] A. Dutta and D. K. Vishwakarma, "Personalized music recommendation system based on streamer streaming trends," in *Proc. 2021 12th Int. Conf. Comput. Commun. Network. Technol. (ICCCNT 2021)*, Kharagpur, India, 2021, doi: 10.1109/ICCCNT51525.2021.9580113.

[3] M. Schedl, P. Knees, B. McFee, and D. Bogdanov, "Music recommendation systems: techniques, use cases, and challenges," in *Recommender Systems Handbook*, 3rd ed. pp. 927–971, 2019, doi: 10.1007/978-1-0716-2197-4_24.

[4] S. Mutturaj, Swetha B., Sangetha P., S. Beldale, and Sahana V., "A survey on hybrid recommendation engine for businesses and users," *Int. J. Inf. Eng. Electron. Bus.*, vol. 13, no. 3, pp. 22–29, Jun. 2021, doi: 10.5815/ijieeb.2021.03.03.

[5] J. Van Balen and B. Goethals, "High-dimensional sparse embeddings for collaborative filtering," in *Proc. World Wide Web. Conf. (WWW 2021)*, vol. 2, pp. 575–581, 2021, doi: 10.1145/3442381.3450054.

[6] N. Ula, C. Setianingsih, and R. A. Nugrahaeni, "Sistem rekomendasi lagu dengan metode content-based filtering berbasis website," (in Indonesian), *e-Proc. Eng.*, vol. 8, no. 6, pp. 12193–12199, Dec. 2021. [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/17229

[7] M. Kaminskas and F. Ricci, "Contextual music information retrieval and recommendation: state of the art and challenges," *Comput. Sci. Rev.*, vol. 6, no. 2–3, pp. 89–119, 2012, doi: 10.1016/j.cosrev.2012.04.002.

[8] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process Syst. (NISP 2017)*, in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5999–6009. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[9] H. Anushree, "Efficient recommendation system using BERT technology," vol. 12, no. 2, pp. 491–500, 2021, doi: 10.34218/IJARET.12.2.2020.047.

[10] M. H. Abdi, G. Okeyo, and R. W. Mwangi, "Matrix factorization techniques for context-aware collaborative filtering recommender systems: a survey," *Comput. Inf. Sci.*, vol. 11, no. 2, pp. 1–10, 2018, doi: 10.5539/cis.v11n2p1.

[11] B. Hawashin, A. Mansour, T. Kanan, and F. Fotouhi, "An efficient cold start solution based on group interests for recommender systems," in *Proc, 1st Int. Conf. Data Sci. E-learning Inf. Syst. (DATA '18),* New York, USA, 2018, Art. no. 26, doi: 10.1145/3279996.3280022.

[12] Y. Deldjoo, M. Schedl, P. Cremonesi, and G. Pasi, "Recommender systems leveraging multimedia content," *ACM Comput. Surv. (CSUR)*, vol. 53, no. 5, 2020, Art. no. 106, doi: 10.1145/3407190.

[13] Y. Zhang, "Music recommendation system and recommendation model based on convolutional neural network," *Mobile Inf. Syst.*, vol. 2022, no. 1, Jan. 2022, Art. no. 3387598, doi: 10.1155/2022/3387598.

[14] M. Schedl, "Deep learning in music recommendation systems," *Front. Appl. Math. Stat.*, vol. 5, Aug. 2019, Art. no. 457883, doi: 10.3389/FAMS.2019.00044.

[15] M. G. Galety, R. Thiagarajan, R. Sangeetha, L. K. B. Vignesh, S. Arun, and R. Krishnamoorthy, "Personalized music recommendation model based on machine learning," in *Proc. 8th Int. Conf. Smart Struct. Syst., (ICSSS 2022)*, Chennai, India, 2022, doi: 10.1109/ICSSS54381.2022.9782288.

[16] A. Elkahky, Y. Song, and X. He, “A multi-view deep learning approach for cross domain user modeling in recommendation systems,” in *Proc. 24th Int. Conf. World Wide Web (WWW 2015)*, May 2015, pp. 278–288, doi: 10.1145/2736277.2741667.
[17] H. Zhu, Y. Niu, D. Fu, and H. Wang, “MusicBERT: a self-supervised learning of music representation,” in *Proc. 29th ACM Int. Conf. Multimedia (MM 2021)*, pp. 3955–3963, Oct. 2021, doi: 10.1145/3474085.3475576.
[18] M. A. Khder, “Web scraping or web crawling: state of art, techniques, approaches and application,” *Int. J. Advance Soft Compu. Appl.*, vol. 13, no. 3, pp. 144–168, Nov. 2021, doi: 10.15849/IJASCA.211128.11.
[19] C. M. Dupin and G. Borglin, “Usability and application of a data integration technique (following the thread) for multi- and mixed methods research: a systematic review,” *Int. J. Nurs. Stud.*, vol. 108, Aug. 2020, Art. no. 103608, doi: 10.1016/J.IJNURSTU.2020.103608.
[20] M. Vystrčilová and L. Peška, “Lyrics or audio for music recommendation?,” in Proc. *10th Int. Conf. Web Intell. Min. Semant. (WIMS 2020)*, New York, USA, pp. 190–194, doi: 10.1145/3405962.3405963.
[21] D. C. Corrales, J. C. Corrales, and A. Ledezma, “How to address the data quality issues in regression models: a guided process for data cleaning,” *Symmetry,* vol. 10, no. 4, Apr. 2018, Art. no. 99, doi: 10.3390/SYM10040099.
[22] K. Dekker and R. Van Der Goot, “Synthetic data for English lexical normalization: how close can we get to manually annotated data?,” in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, 2020, pp. 6300–6309. [Online]. Available: https://aclanthology.org/2020.lrec-1.773/
[23] D. J. Ladani and N. P. Desai, “Stopword identification and removal techniques on TC and IR applications: a survey,” in *Proc. 2020 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS 2020)*, Coimbatore, India, Mar. 2020, pp. 466–472, doi: 10.1109/ICACCS48705.2020.9074166.
[24] D. Khyani, B. S. Siddhartha, N. M. Niveditha, and B. M. Divya, “An interpretation of lemmatization and stemming in natural language processing,” *J. Univ. Shanghai Sci. Technol.* vol. 22, no. 10, pp. 350–357, 2021. [Online]. Available: https://jusst.org/an-interpretation-of-lemmatization-and-stemming-in-natural-language-processing/
[25] M. O. Sanjaya, S. Bukhori, and M. Furqon, “Virtual assistant for thesis technical guide using artificial neural network,” *Indonesian J. Artif. Intell. Data Min.*, vol. 6, no. 2, pp. 188–196. [Online]. Available: https://ejournal.uin-suska.ac.id/index.php/IJAIDM/article/view/23473
[26] Y. Zhang *et al.*, “DIALOGPT: large-scale generative pre-training for conversational response generation,” in *Proc. Ann. Meet. Assoc. Comput. Linguistics*, 2020, pp. 270–278, doi: 10.18653/v1/2020.acl-demos.30.
[27] T. Maheshwari, T. N. Bhaveshbhai, and M. Halder, “The power of visual analytics and language processing to explore the underlying trend of highly popular song lyrics,” *Eng. Appl. Sci. Lett.*, vol. 4, no. 3, pp. 19–29, 2021, doi: 10.30538/psrp-easl2021.0072.
[28] Y. Song, S. Dixon, and M. T. Pearce, “A survey of music recommendation systems and future perspectives,” in *Proc. 9th Int. Symp. Comp. Music Model. Retrieval (CMMR)*, London, UK, 2012, pp. 395–410. [Online]. Available: https://www.eecs.qmul.ac.uk/~simond/pub/2012/Song-Dixon-Pearce-CMMR-2012.pdf
[29] U. Ungkawa, D. Rosmala, and F. Aryanti, “Pembangunan aplikasi travel recommender dengan metode case base reasoning,” (in Indonesian) *J. Informatika*, vol. 4, no. 2, pp. 57–68, 2013. [Online]. Available: https://lib.itenas.ac.id/kti/wp-content/uploads/2013/10/Jurnal-No.2-vol.4-3.pdf
[30] N. Gali and V. Tiwari, “Speech and lyric-based Doc2Vec music recommendation system,” *Int. J. Eng. Res. Technol. (IJERT)*, vol. 9, no. 8, pp. 67–70, 2021, doi: 10.17577/IJERTCONV9IS08015.
[31] S. Ebiesuwa *et al.* “Analysis of Nigeria’s top ten song lyrics using natural language processing techniques,” *Intell. Syst. Appl. Eng.*, vol. 12, no. 4, pp. 4431–4438, 2024.
[32] A. W. Qurashi, V. Holmes, and A. P. Johnson, “Document processing: methods for semantic text similarity analysis,” in *Proc. 2020 Int. Con. Innov. Intell. Syst. Appl. (INISTA 2020)*, Novi Sad, Serbia, Aug. 2020, pp. 1–6 doi: 10.1109/INISTA49547.2020.9194665.
[33] J. Wang and Y. Dong, “Measurement of text similarity: a survey,” *Information*, vol. 11, no. 9, Aug. 2020, Art. no. 421, doi: 10.3390/INFO11090421.
[34] B. V. Kartika, M. J. Alfredo, and G. P. Kusuma, “Fine-tuned IndoBERT based model and data augmentation for Indonesian language paraphrase identification.,” *Revue d’Intelligence Artificielle*, vol. 37, no. 3, pp. 733–743, Jun 2023, doi: 10.18280/ria.370322.
[35] W. Wongso, H. Lucky, and D. Suhartono, “Pre-trained transformer-based language models for sundanese,” *J. Big Data*, vol. 9, no. 1, Dec. 2022, Art. no. 39 doi: 10.1186/S40537-022-00590-7.
[36] L. Tunstall, L. Von Werra, and T. Wolf, *Natural Language Processing with Transformers*,O’Reilly Media Inc.2022.
[37] S. Wehnert, V. Sudhi, S. Dureja, L. Kutty, S. Shahania, and E. W. De Luca, “Legal norm retrieval with variations of the BERT model combined with TF-IDF vectorization,” in *Proc.18th Int.Conf.Artif.Intell.Law (ICAIL 2021)*, New York, USA, Jun. 2021, doi: 10.1145/3462757.3466104.
[38] J. E. Ewusie, C. Soobiah, E. Blondal, J. Beyene, L. Thabane, and J. S. Hamid, “Methods, applications and challenges in the analysis of interrupted time series data: a scoping review,” *J. Multidiscip. Healthc.*, vol. 13, pp. 411–423, 2020, doi: 10.2147/JMDH.S241085.
[39] Q. C. Song, C. Tang, and S. Wee, “Making sense of model generalizability: a tutorial on cross-validation in R and Shiny,” *Adv. Methods Pract. Psychol. Sci.*, vol. 4, no. 1, Mar. 2021, doi: 10.1177/2515245920947067.
[40] V. R. Revathy, A. S. Pillai, and F. Daneshfar, “LyEmoBERT: classification of lyrics’ emotion and recommendation using a pre-trained model,” *Procedia. Comput. Sci.*, vol. 218, pp. 1196–1208, Jan. 2023, doi: 10.1016/J.PROCS.2023.01.098.
[41] A. Elbir, H. Bilal Çam, M. E. Iyican, B. Öztürk, and N. Aydin, “Music genre classification and recommendation by using machine learning techniques,” in *Proc. 2018 Innov. Intell. Syst. Appl. Conf. (ASYU 2018)*, Nov. 2018, pp. 1–5, doi: 10.1109/ASYU.2018.8554016.
[42] C. McDonald, A. E. Foster, and P. Rafferty, “Playlists and genre: the role of music genre in Spotify’s playlists,” *J. Documentation*, vol. 81, no.1, pp. 1–23, 2024, doi: 10.1108/JD-08-2023-0152.