PAPER NAME	AUTHOR
atharv plag check.docx	athrav g
WORD COUNT	CHARACTER COUNT
1906 Words	11395 Characters
PAGE COUNT	FILE SIZE
10 Pages	145.3KB
SUBMISSION DATE	REPORT DATE
Jul 3, 2023 11:31 AM GMT+5:30	Jul 3, 2023 11:31 AM GMT+5:30

● 5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 1% Internet database
- 1% Publications database
- Crossref database
- Crossref Posted Content database
- 5% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Small Matches (Less then 8 words)

ABSTRACT

The creation and implementation of a Python-based song recommendation system which makes use of AI filtering approaches are presented in this study. The system's goal is to offer consumers personalised music recommendations based on their listening habits and interests.
The collaborative filtering and content-based filtering techniques are combined in the recommendation algorithm to increase the recommendations' relevance and accuracy.
Data preprocessing methods are applied to clean and transform the collected song data. The system incorporates AI filtering mechanisms to analyse user behaviour, identify patterns, and generate customized recommendations.
Integration with external APIs, such as Spotify, allows access to a wide range of music, enhancing the recommendation process.
The system's performance is evaluated using appropriate metrics, demonstrating its effectiveness in delivering personalized and satisfactory song recommendations.
This project highlights the capabilities of Python and AI techniques in building intelligent song recommendation systems that enhance user engagement and satisfaction in the music domain.

Introduction

The music Recommendation System is an effective tool that uses machine learning methodologies to offer consumers personalised music choices based on their tastes and listening habits. The system's capacity to improve user experience by providing personalised music selections has attracted a lot of attention. In this research, shared filtering, [7] content-based filtering, as well as hybrid algorithms are investigated as potential approaches to build an efficient music recommendation system.

In order to produce recommendations, collaborative filtering concentrates on user behaviour trends and user commonalities. It examines the listening preferences and habits of users with comparable tastes and recommends music that are relevant to their interests. Contrarily, content-based filtering uses song elements including style, artist, song lyrics, and audio properties to suggest songs that are comparable to songs that a user has previously appreciated. In this method, the recommender functions as a classifier that learns the user's preferences from song attributes. Recommendation is treated as a specific to the user classification problem.

An extensive and diversified dataset is necessary to train the music recommendation algorithm. You can use a variety of publically accessible datasets, such the Million Song Dataset, that consist of audio attributes, metadata, including user interactions. This dataset offers a plethora of data that can be used to efficiently train and test recommendation models.

The functionality and user base of the melody recommendation system can be significantly increased by integrating it with well-known music streaming services such as Spotify and Apple Music. The recommendation system can provide more precise and individualised recommendations by making use of the enormous collection of music and user data made available by these sites. The easy access to a variety of songs made possible by the integration guarantees that customers are given relevant and entertaining music options.

The effectiveness of the song recommendation system can be evaluated through metrics such as precision, recall, and user satisfaction. By conducting thorough evaluations and comparisons with baseline models, we can measure the system's performance and assess its ability to provide relevant and engaging song recommendations.

In conclusion, the creation of a music recommendation system that makes use of techniques like shared filtering and filtering based on content can significantly improve consumers' musical enjoyment. By leveraging diverse datasets and integrating with popular music streaming platforms, such as Spotify and Apple Music, the system can offer personalized recommendations that align with users' preferences, resulting in improved user satisfaction and engagement.

Literature Review

Due to the rising popularity of personalised music streaming services, the area of song recommendation algorithms has made tremendous strides in recent years. This study of the literature seeks to present an overview of the many strategies and methods utilised in creating efficient music recommendation systems, in addition to the evaluation methodologies applied to judge their effectiveness.

Collaborative filtering:

It is a strategy that many song recommendation systems employ. The generation of recommendations is based on the preferences of all users. Early research demonstrated the value of collective filtering by showcasing how user-based suggestions may be formed from user ratings & feedback, such as Resnick and Varian's work from 1997. More recent research has focused on overcoming the limitations of shared filtering, such as: Sparsity Problem and the Cold-Start Problem, through techniques like matrix factorization and neighborhood-based methods (Koren et al., 2009; Sarwar et al., 2001).

To provide individualised music recommendations, user behaviour and tastes are analysed. Collaborative filtering algorithms may generate music recommendations that are in line with a user's musical tastes and preferences by seeing trends and commonalities throughout users' interactions with songs.

Both user-based group filtering and item-based joint filtering, which are two distinct types of collaborative filtering, employ users' collective wisdom to generate recommendations. Furthermore, more advanced techniques, such as factorization by matrix, have been applied to circumvent issues like sparsity and the cold-start problem. The following are the primary factors that collaborative filtering in audio recommendation systems should take into account:

1. For creating customised song recommendations, collaborative filtering is an established method.
2. In order to find trends and similarities, it examines user behaviour and preferences.
3. Typical methods consist of item-based collaborative filtering and user-based collaborative filtering.
4. An improved method used to overcome problems in collaborative filtering is matrix factorization.
5. By offering pertinent and interesting music suggestions, collaborative filtering improves the user experience.

Content-based filtering:

It has also attracted interest from music suggestion programmes. This method focuses on using song elements including style, creator, lyrics, & acoustic qualities to find commonalities and propose songs appropriately. McFee et al. (2012) explored the use of audio content analysis to extract features like timbre, rhythm, and melody, and demonstrated its effectiveness in generating song recommendationsIn addition, Li et al.

(2019) suggested a hybrid strategy that combines collaborative filtering with content-based filtering to take advantage of the complimentary qualities of both approaches.

Approach:

In order to increase recommendation accuracy and solve the shortcomings of individual approaches, flexible systems that combine collaborator filtered and material filtered results have been widely utilised. In order to deliver more precise and individualised recommendations, hybrid models—such as those put forth by Baltrunas et al. (2011) & Desrosiers et al. (2011)—combine user-item interaction information with song attributes. These models have the benefit of preserving both the musical features and the individual tastes of similar consumers.

Evaluation:

To guarantee that the suggested methods are effective, it is essential to evaluate the performance of tune recommendation systems. The accuracy, recollection, and Mean Average Precision (MAP), which is are common evaluation criteria. Bell et al. (2009) proposed the use of normalized discounted cumulative gain (nDCG) to evaluate the ranking quality of recommendations. Furthermore, user-centric assessments via online A/B testing or user surveys offer insightful data on user satisfaction and engagement.

Integration

Integration with popular music engaging platforms, such as Spotify and Apple Music, presents opportunities to enhance song recommendation systems. Spotify, for instance, offers a rich collection of user interactions and comprehensive song metadata. The work of McFee et al. (2015) showcased the integration of external data sources, such as Spotify's audio analysis and Last. FM’s listening history, to improve recommendation accuracy. Similarly, Apple Music's integration provides access to a vast music library and user data, enabling personalized recommendations.

Conclusion

From straightforward user-based shared filtering to more complex methods, audio recommendation systems have developed, incorporating content-based filtering and hybrid models. The evaluation of these systems through various metrics and user-centric studies ensures their effectiveness in providing accurate and engaging recommendations. Integration with popular music streaming platforms like Spotify and Apple Music further enhances the capabilities of these systems. In order to produce even more customised and dynamic song recommendations, future research can concentrate on investigating fresh methodologies, tackling issues like the cold-start problem, including incorporating contextual information.

Music recommender system

Recommendation systems are one of the most commonly used machine learning methods. A recommender system, also known as a recommendation engine, is an algorithm for filtering whose goal is to anticipate the assessment or preference that a user would assign to a certain object, such as a movie, a product, a music, etc.

Which type of recommender can we have?

The following are the two main types of recommender systems:

- Content-based filters
- Collaborative filters

Based on what a person has already enjoyed, content-based filters can forecast what they will like.
On the other hand, collaborative-based filters predict what a user like based on what other users, that are similar to that particular user, have liked.

1) Content-based filters

One could consider the recommendations offered by **content-driven recommenders** to have a problem with user-specific categorisation. The properties of the song are used by this classifier to estimate the user's preferences.

Keyword matching is the easiest method to use.

In a nutshell, the concept is to extract significant keywords from an audio description a user loves, look for those keywords in comparable song descriptions, and then offer to the user those tracks.

How is this performed?

The **Terminology Frequent-Inverse Documents Frequencies (TF-IDF)** methodology can be used in our circumstance because of the characteristics of textual input for the purpose of matchmaking activity.

We'll go through the steps for generating a **content-based** music recommender system.

Importing required libraries

First, we'll import all the required libraries.

```
import numpy as np
import pandas as pd
```

```
from typing import List, Dict
```

The TF-IDF score has already been employed by us for Twitter sentiment analysis.

Similarly, we will once more utilise TfidfVectorizer that comes in the Scikit Learn package..

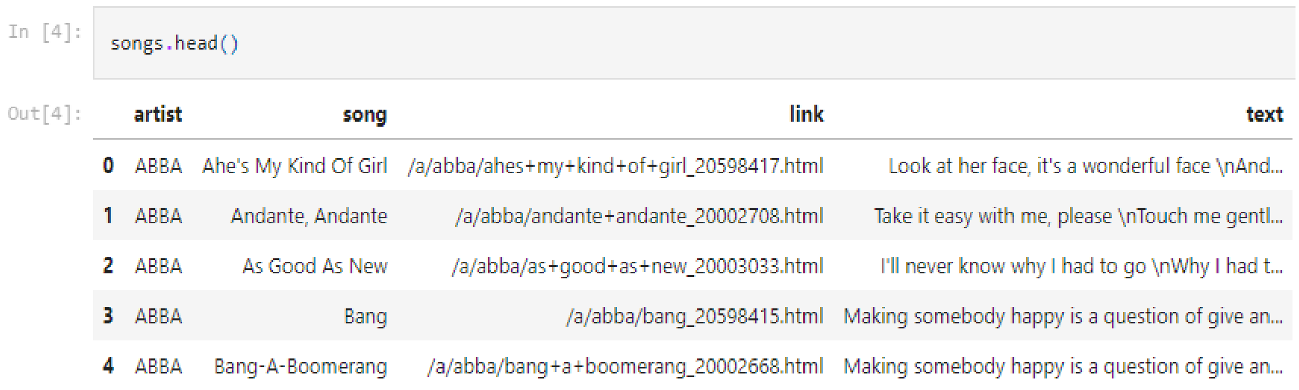
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Dataset

Consider that we've got the following set of data.

Creator Tags, Performer, & Phrases of 57650 English songs are included in this collection. Scraping was used to obtain the data from **LyricsFreak**.

```
songs = pd.read_csv('content based recommedation system/songdata.csv')
```



```
In [4]: songs.head()
```

Out[4]:

	artist	song	link	text
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face \nAnd...
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please \nTouch me gentl...
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I had to go \nWhy I had t...
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happy is a question of give an...
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happy is a question of give an...

The dataset is so large that we will only resample 5000 random songs.

```
songs = songs.sample(n=5000).drop('link', axis=1).reset_index(drop=True)
```

We also see that there is a n in the content, therefore we will eliminate it.

```python
songs['text'] = songs['text'].str.replace(r'\n', '')
```

The TF-IDF vectorizer is then used to determine the TF-IDF score for each song lyric, word by word.

In this instance, we pay close attention to the parameters that we can specify.

```python
tfidf = TfidfVectorizer(analyzer='word', stop_words='english')
```

```python
lyrics_matrix = tfidf.fit_transform(songs['text'])
```

How can we make a recommendation using this matrix?

Now, we must determine how closely two lyrics resemble one another. We'll employ **cosine similarity**.

Each item's cosine similarity to all other items in the set of data will be determined

Therefore, we simply provide lyrics_matrix like a parameter..

```python
cosine_similarities = cosine_similarity(lyrics_matrix)
```

50 songs that are most similar to every song within our dataset will be listed in a dictionary after we have the similarities.

```python
similarities = {}
```

```python
for i in range(len(cosine_similarities)):
    # Now we'll sort each element in cosine_similarities and get the indexes of the songs.
    similar_indices = cosine_similarities[i].argsort()[:-50:-1]
    # After that, we'll store in similarities each name of the 50 most similar songs.
    # Except the first one that is the same song.
    similarities[songs['song'].iloc[i]] = [(cosine_similarities[i][x], songs['song'][x], songs['artist'][x]) for x in similar_indices][1:]
```

After that, all the magic happens. We may locate the most comparable things using those similarity scores and make a recommendation.

For that, we'll instantiate our Content driven suggestion class.

```python
class ContentBasedRecommender:
    def __init__(self, matrix):
        self.matrix_similar = matrix

    def _print_message(self, song, recom_song):
        rec_items = len(recom_song)

        print(f'The {rec_items} recommended songs for {song} are:')
        for i in range(rec_items):
            print(f"Number {i+1}:")
            print(f"{recom_song[i][1]} by {recom_song[i][2]} with {round(recom_song[i][0], 3)} similarity score")
            print("--------------------")

    def recommend(self, recommendation):
        # Get song to find recommendations for
        song = recommendation['song']
        # Get number of songs to recommend
        number_songs = recommendation['number_songs']
        # Get the number of songs most similars from matrix similarities
        recom_song = self.matrix_similar[song][:number_songs]
        # print each item
        self._print_message(song=song, recom_song=recom_song)
```

Thus, we Instantiate the class.

```python
recommedations = ContentBasedRecommender(similarities)
```

After that, we are prepared to choose some music from the dataset & offer a suggestion.

```python
recommendation = {
    "song": songs['song'].iloc[10],
    "number_songs": 4
}
```

```python
recommedations.recommend(recommendation)
```

Output:

```
The 4 recommended songs for Jehovah and All That Jazz are:
Number 1:
Sing by Glen Campbell with 0.166 similarity score
--------------------
Number 2:
Devil Cried by Black Sabbath with 0.149 similarity score
--------------------
Number 3:
Angelique by Kenny Loggins with 0.141 similarity score
--------------------
Number 4:
Up The Devil's Pay by Old 97's with 0.131 similarity score
--------------------
```

We have the option to select another random piece of music for further suggestion.

```
recommendation2 = {
    "song": songs['song'].iloc[120],
    "number_songs": 4
}
```

```
recommedations.recommend(recommendation2)
```

Output:

```
The 4 recommended songs for I Do It For Your Love are:
Number 1:
I Love You by Lionel Richie with 0.189 similarity score
--------------------
Number 2:
Just One Love by Michael Bolton with 0.187 similarity score
--------------------
Number 3:
I'm Gonna Sit Right Down and Write Myself A Letter by Nat King Cole with 0.
184 similarity score
--------------------
Number 4:
If I Love Again by Barbra Streisand with 0.183 similarity score
--------------------
```

● 5% Overall Similarity

Top sources found in the following databases:

- 1% Internet database
- 1% Publications database
- Crossref database
- Crossref Posted Content database
- 5% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Liverpool John Moores University on 2023-06-12 Submitted works	2%
2	University of Northumbria at Newcastle on 2023-05-15 Submitted works	<1%
3	Queen Mary and Westfield College on 2023-04-25 Submitted works	<1%
4	pdfcoffee.com Internet	<1%
5	ee.columbia.edu Internet	<1%
6	Victoria University on 2023-07-01 Submitted works	<1%
7	Victoria University on 2023-07-01 Submitted works	<1%