MUSIC RECOMMENDATION SYSTEM WITH PLAGIARISM DETECTION

Mr. P. Krishnanjaneyulu[1], Sheema Patro[2], V. Jahnavi[2], P. N. V. S Dhanush[2], G. Mahesh[2]

[1]Assistant Professor at Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India
[2]Final year students of Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

Abstract— In this paper, we present a personalized music recommendation system based on the KNN and machine learning algorithms. In personalized music recommendation system, we propose a collaborative filtering and content filtering recommendation algorithm here we use log file which stores the recommendations to user . The proposed system contains the log files which stores the previous history of playlist of music by the user. Here in this paper we extracts data of the users access files from the history of logs and recommend them. Content-based methods gives recommendations based on the similarity of two song contents or attributes while here we implement the matrix of different songs and perform the collaborative filtering methods on different songs for recommendation system The plagiarism system extracts the music from input and finds music that are close to the query music which the query has plagiarized. We use the million song dataset to evaluate the personalized music recommendation system. The data cleaning is done by the data science algorithms. The plagiarism detection is done by finding the similar music genre which minimizes the issue of copyrights.

Index Terms— collaborative filtering algorithm. KNN, cosine similarity, tf-idf, CSR Matrix

Introduction

With the growth of the internet in recent decades, it has become the primary source for retrieving multimedia material such as video, literature, and music, among other things. People regard music to be a significant part of their lives, and they listen to it on a regular basis. The issue now is how to organize and manage the millions of music titles that society produces. A smart music recommendation system should be able to detect preferences automatically and produce playlists based on them. The suggested technique uses music to detect plagiarism in music similarity.

The plagiarism algorithm takes music from input and discovers music that is similar to the query music that has been plagiarised by the query. Meanwhile, the advent of recommender systems gives the music industry a fantastic opportunity to gather users who are interested in music. Using KNN and Machine Learning, we need to create the finest music recommendation system that can forecast depending on customization.

The collaborative filtering algorithm has been confirmed to function well based on user listening behavior and historical ratings. The Music Information Retrieval Evaluation Exchange (MIREX) has been organized annually since 2005 to aid in the development of MIR algorithms. Similarity between songs is assessed in the content-based method, and songs are recommended based on the similarity score. Plagiarism here we can use content based filtering techniques; plagiarism can be recognized based on similarity. Score.

We may quickly propose music to users based on their interests and moods using mood prediction. The mood can be predicted in a variety of methods, including using lyrics, face expression detection, and so on. In this case, we utilized lyrics-based mood prediction, which calculates the similarity score and recommends music.

Music recommendation systems are a two-edged sword. They are advantageous to both the user and the provider. They keep the user engaged by providing fascinating music in the form of suggestions, minimizing the number of options available to the user. They allow for the investigation and discovery of music that the user might not be aware of. There is never a lack of enjoyment because it is a music recommender.

2. Related Work

Existing collaborative filtering algorithms-based recommender systems have had a lot of success. Netflix held a competition for the best collaborative filtering algorithm [3], and the algorithm, here it implements latent factor models, which is improved by 8.1 percent. Amazon employs user-to-user and item-to-item collaborative filtering [4], which is critical to the company's success. A fresh neural network-based algorithm, neural collaborative filtering (He 2017) [5, has just been proposed. Many academics have proposed several methods employing Machine Learning techniques for content-based algorithms, such as Decision Tree Algorithm based was implemented here for the recommendation system [6], the algorithms like SVM was used in this paper [7], and even logistic regression [8]. To construct these algorithms, we may fully utilize the knowledge we gained in class. While the music recommendation system resembles existing commercial recommendation systems in certain ways, it concentrates on offering good and individualized music advise rather than things for users to purchase. The ideal music recommendation system would be able to make individualized music recommendations to human listeners automatically. The length of a piece of music is significantly less than that of a book or a movie, and people usually listen to the songs they like more and more times which is a challenging task in implementation of the recommendation system

3 Proposed Work

1. Collaborative Filtering

2. Plagiarism and content-based module

3. Mood Prediction

Algorithm: Collaborative filtering

3.1 KNN

The goal of this method is to develop a function that can predict whether or not a user would profit from an item — in this case, whether or not the user would listen to a music. This can be accomplished through the use of ratings. User ratings can be collected in two ways: explicitly and implicitly. The K-Nearest Neighbors Algorithm was utilized.

3.2 Explicit Rating:

This means we explicitly ask the user to give a rating. This represents the most direct feedback from users to show how much they like a song.

3.3 Implicit Rating:

We examine whether or not a user listened to a song, for how long or how many times, which may suggest that he/she liked that particular song.

3.4 Interaction matrices

These are based on the many entries that include a user-song pair as well as a value that represents the user's rating for that song.

3.5 KNN:

K - Nearest Neighbors (KNN) is considered the standard method when it comes to both user-based and item-based collaborative filtering approaches. The KNN algorithm is a supervised non-parametric Lazy Learning method used for both classification and regression. It considers a graph based on the rating and plots the rating of the input song in the graph and calculates the distance with all the other songs and recommends the song based on the distance I.e. least distance is compared.

3.6 CSR Matrix:

Sparse matrix in general are collections in which vast majority of values are some default values usually none or 0.CSR stands for "Compressed Sparse Row". The CSR notation output the row-column tuple where the matrix contains non-zero values along with those values. It gives the information of the listen count and its location other than 0 in music recommendation system.

3.7 Fuzzy Wuzzy:

Fuzzywuzzy is a string matching library written in Python. The process of matching strings that match a specified pattern is known as fuzzywuzzy. The difference between sequences is calculated using Levenshtein distance. It compares the two strings and returns a similarity index. If we provide an incorrect song, it compares it to other songs, determines the similarity using the fuzzy matching function, calculates the ratio, and uses the song with the highest ratio as the input song.

3.8 Plagiarism:

3.8.1 Cosine similarity (Content-Based Model)

The following two phases must be completed by a content-based recommendation system. To begin, extract features from the song descriptions' content to generate an object representation. Second, create a similarity function among these object representations that resembles the item-item similarity that humans recognize. Because we're dealing with text and words, we may use Term Frequency-Inverse Document Frequency (TF-IDF) to match them.

3.8.2 TF-IDF

It is a methodology for retrieving information that considers the terms frequency (TF) and inverse document frequency (IDF). The TF*IDF weight of a phrase is defined as the product of these scores. The TF-Idf score indicates how uncommon a phrase is in a given document, and vice versa. We'll use cosine similarity, namely its implementation in Scikit-learn, for our song recommendation engine. We want to see how similar each item is to every other item in the collection using cosine similarity. As a result, we just supply the lyrics matrix as an argument. Once we get the similarities, we'll save the names of the 50 most similar songs for each song in our dataset in a dictionary named similarities.

The metric of cosine similarity is used to determine how similar two objects are. It estimates the cosine of the angle formed by two vectors projected onto a multi-dimensional space mathematically. The output value is between 0 and 1. A value of 0 indicates no similarity, whereas a value of 1 indicates that both objects are identical. The cosine similarity in Python is calculated as the dot product of the input samples.

3.9 Mood Prediction

The digitization of music has made various types of music more accessible to people all over the world. Increased work pressure takes away the time needed to listen to and assess music in order to build a personal music library. One option could be to create a mood-based music search engine or recommendation system. Combining a wide range of semantic and stylistic elements collected from textual lyrics, develop a mood classification system.

3.10 Count Vectorizer:

The CountVectorizer in Scikit-learn is used to turn a set of text documents into a vector of term/token counts. It also allows text data to be pre-processed before being converted into a vector format. Because of this, it's a text feature representation module with a lot of flexibility.

3.11 Fit_transform():

The fit transform() function is applied to the training data in order to scale it and learn its scaling parameters. Here, the model we developed will learn the mean and variance of the training set's characteristics. Our test data is then scaled using the parameters we've learned.

3.12 TfidfVectorizer Model:

Tfidf Vectorizer - Converts text into feature vectors that can be used as estimator input. vocabulary_ is a dictionary that translates each token (word) into a feature index in the matrix, with a feature index for each unique token. The integers (weights) in each vector indicate the tf-idf score features. Tfidf transformer and Tfidf vectorizer from Scikit-learn try to perform the same thing: convert a collection of raw documents into a matrix of TF-IDF features.

3.13 Model:

N items in a continuous sequence from a given sample of text or speech." A character, a word, or a sentence can be used as an item, and N can be any integer. When N is 2, the sequence is referred to as a bigram. A trigram, for example, is a sequence of three things, and so on. In the form of a (n 1)–order Markov model, an n-gram model is a sort of probabilistic language model for predicting the next item in a sequence.

3.14 SVM:

SVM (Support Vector Machine) is a supervised machine learning technique that can be used to solve classification and regression problems. It is, however, mostly employed to solve categorization difficulties. Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a certain coordinate in the SVM algorithm. The Support Vector Machine, or SVM, is a linear model that can be used to solve classification and regression issues. It can solve both linear and nonlinear problems and is useful for a wide range of applications. SVM is a basic concept: The method divides the data into classes by drawing a line or hyperplane. During the training phase, the training dataset divides the words into different categories such as happy, sad, and so on, and based on the words, the mood of the input song is predicted, and based on the similarity score, which is sorted in increasing order, the mood for the top similarity songs is predicted using SVM, and then similar mood songs are recommended.

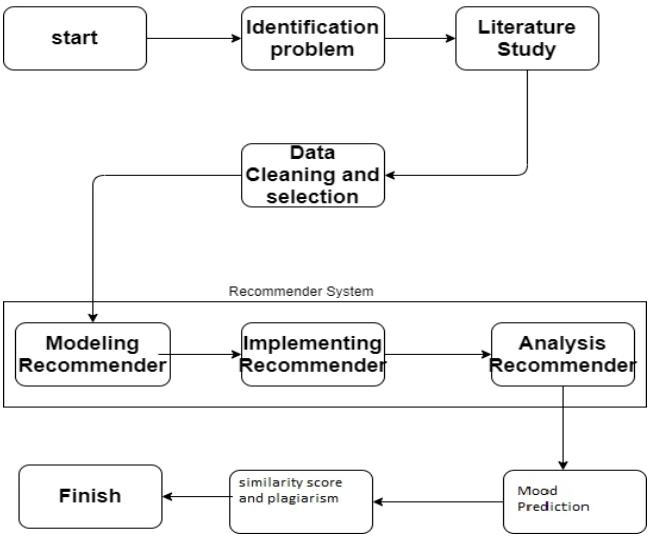


Fig 1: Steps of music recommendation system

4. Algorithms

4.1 KNN

The K Nearest Neighbor algorithm is a type of supervised learning technique that is used for classification and regression. It's a flexible approach that may also be used to fill in missing values and resample datasets. K Nearest Neighbor considers K Nearest Neighbors (Data points) to predict the class or continuous value for a new Datapoint, as the name suggests

1) Instance-based learning uses full training instances to predict output for unknown data, rather than learning weights from training data to predict output (as in model-based algorithms).

2) Lazy Learning: The model is not learned using training data before the prediction is required on the new instance, and the learning process is postponed until the prediction is asked.

3) Non-Parametric: In KNN, the mapping function has no specified form.

k-Nearest Neighbor
Classify (**X**, **Y**, *x*) // **X**: training data, **Y**: class labels of **X**, *x*: unknown sample
for $i = 1$ to m **do**
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set I containing indices for the k smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$

Fig 2: Steps for KNN algorithm implementation

4.2 Cosine similarity

If you can determine whether two things are similar, you can utilize that information to construct more sophisticated jobs, such as:

4.2.1 Search: find the most similar document to a given one

4.2.2 Classification: is some customer likely to buy that product

4.2.3 Clustering: are there natural groups of similar documents

4.2.4 Product recommendations: which products are similar to the customer's past purchases

```
int LevenshteinDistance(char S[1..M], char T[1..N]){
    declare int d[0..M, 0..N]
    for i from 0 to M
      d[i,0] := i //the distance of any first string to an empty second string
    for j from 0 to N
      d[0, j] := j //the distance of any second string to an empty first string
    for j from 1 to N
    {
      for i from 1 to M
      {
        if S[i] = T[j] then
          d[i, j] := d[i-1, j-1] //no operation required
        else d[i, j] := minimum(d[i-1, j] + 1, //a deletion
                                d[i, j-1] + 1, //an insertion
                                d[i-1, j-1] + 1) //a substitution
      }
    }
    return d[M, N]
}
```

Fig3: algorithm implementation for cosine similarity

5. Performance Analysis

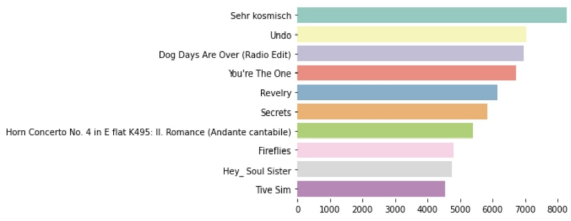


Fig 4 Graph between title and listen count

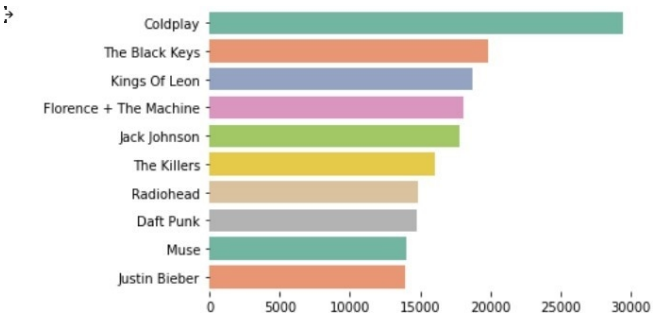


Fig 5 Graph between artist name and listen

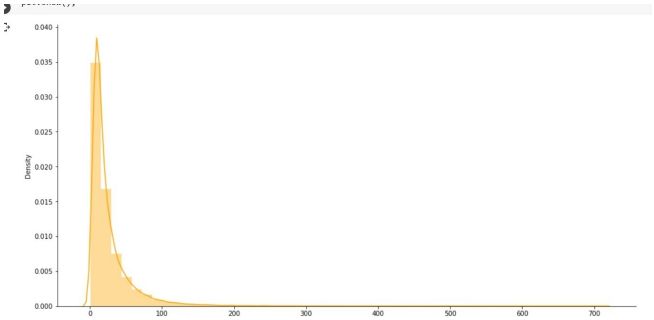
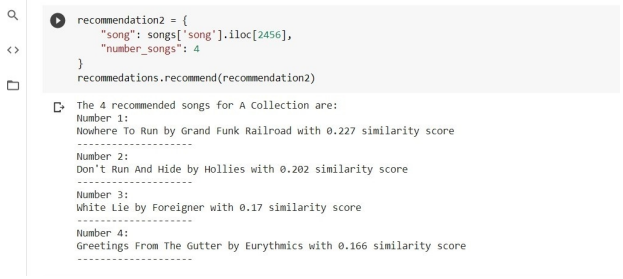


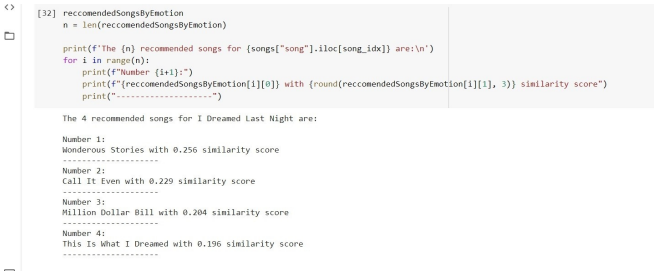
Fig 6: listen count graph by each user



```
recommendation2 = {
    "song": songs['song'].iloc[2456],
    "number_songs": 4
}
recommedations.recommend(recommendation2)
```

```
The 4 recommended songs for A Collection are:
Number 1:
Nowhere To Run by Grand Funk Railroad with 0.227 similarity score
--------------------
Number 2:
Don't Run And Hide by Hollies with 0.202 similarity score
--------------------
Number 3:
White Lie by Foreigner with 0.17 similarity score
--------------------
Number 4:
Greetings From The Gutter by Eurythmics with 0.166 similarity score
--------------------
```

Fig 7: Content based recommendation system



```
[32] reccomendedSongsByEmotion
     n = len(reccomendedSongsByEmotion)

     print(f'The {n} recommended songs for {songs["song"].iloc[song_idx]} are:\n')
     for i in range(n):
         print(f"Number {i+1}:")
         print(f"{reccomendedSongsByEmotion[i][0]} with {round(reccomendedSongsByEmotion[i][1], 3)} similarity score")
         print("--------------------")

     The 4 recommended songs for I Dreamed Last Night are:

     Number 1:
     Wonderous Stories with 0.256 similarity score
     --------------------
     Number 2:
     Call It Even with 0.229 similarity score
     --------------------
     Number 3:
     Million Dollar Bill with 0.204 similarity score
     --------------------
     Number 4:
     This Is What I Dreamed with 0.196 similarity score
     --------------------
```

Fig 8: emotion detection in songs

Conclusion

The following are our conclusions based on experiment results. First, music recommender system should consider the music genre information to increase the quality of music recommendations. The music recommender is able to recommend the songs based on the song features. The music Recommender is able to check plagiarism in the dataset taken by generating the similarity score for each recommended song. The mood of the song is predicted by examining the lyrics of the given song with all the other songs in the dataset and predicting the mood and similarity scores and recommending the songs based on the mood. The complex nature of the machine learning systems like the Music Recommendation System can't have a standardized structure because different music recommender systems work in different way. Based on our analyses, we can suggest for future research to add other music features in order to improve the accuracy of the recommender system,

such as using tempo gram for capturing local tempo at a certain time.

References

[1] Everyone listens to music, but how we listen is changing. [online] Available at: http://www.nielsen.com/us/en/insights/news/2015/everyone-listens-to-music-but-how- we-listen-is-changing.html [Accessed 10 Oct. 2017].

[2] Labrosa.ee.columbia.edu. (2017). Million Song Dataset | scaling MIR research. [online] Available at: https://labrosa.ee.columbia.edu/millionsong/ [Accessed 10 Oct. 2017].

[3] en.wikipedia.org. (2017). Netflix Prize. [online] Available at: https://en.wikipedia.org/wiki/Netflix_Prize [Accessed 11 Oct. 2017].

[4] Linden, G., Smith, B. and York, J. (2003). Amazon.com Recommendations Item-to- Item Collaborative Filtering. [ebook] Available at: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf [Accessed 10 Oct. 2017].

[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 173- 182. DOI: https://doi.org/10.1145/3038912.3052569

[6] Gershman, A. and Meisels, A. (2015). A Decision Tree Based Recommender System. [ebook] IEEE Xplore, pp.170-179. Available at: https://subs.emis.de/LNI/Proceedings/Proceedings165/170.pdf [Accessed 9 Oct. 2017].

[7] Min SH., Han I. (2005) Recommender Systems Using Support Vector Machines. In: Lowe D., Gaedke M. (eds) Web Engineering. ICWE 2005. Lecture Notes in Computer Science, vol 3579. Springer, Berlin, Heidelberg

[8] R-bloggers. (2017). Hybrid content-based and collaborative filtering recommendations with {ordinal} logistic regression (2): Recommendation as discrete choice. [online] Available at: https://www.r-bloggers.com/hybrid-content-based-and- collaborative-filtering-recommendations-with-ordinal-logistic-regression-2-recommendation-as-discrete-choice/ [Accessed 7 Oct. 2017].