**A PROJECT REPORT**

**ON**

**MUSIC RECOMMENDER SYSTEM**

Submitted in partial fulfillment of the requirement for the

IVth semester

**Bachelor of Technology**

**By**

**ARYAN RAWAT**

**2016680**



**DEPARTMENT OF COMPUTER SCIENCE &** **ENGINEERING**

**GRAPHIC ERA DEEMED UNIVERSITY**

**DEHRADUN**

**2022-2023**

**ACKNOWLEDGEMENT**

Here by I am submitting the project report on **“Music recommender system ”** as per the scheme of Graphic Era Deemed University, Dehradun.

I would like to express our sincere gratitude to Ms. Vishu Tyagi**,** for providing a congenial environment to work in and carry out our project.

Finally, I am very much thankful to all the faculty members of the Department of Computer Science and Technology, friends and our parents for their constant encouragement, support and help throughout the period of project conduction.

NAME:ARYAN RAWAT

ROLL NO:2016680

MOTIVATION

 I have been listening to English hip hop music from the past 7 years. Back then as I had certain restrictions on the internet so I only knew about some artists but as the Internet grew and apps like spotify and souncloud so the accessibility to songs became more and more easier. So there were many choices to c hoose from. As I don’t have enough time to listen to each and every artists so then on my mind , I thought of an idea that how about I make an recommendation system which can recommend me songs on the basis of what type of genre and artists I most listen to. So , hence this was the basic motivation I had which encouraged me to make this project.

**PROBLEM STATEMENT**

**Music Recommender System**

**SOFTWARE USED**

* Jupyter notebook,Anaconda

**TOOLS USED**

* Jupyter notebook libararies like pandas,numpy,sklearn,etc.

**LANGUAGE**

* Python

**INTRODUCTION**

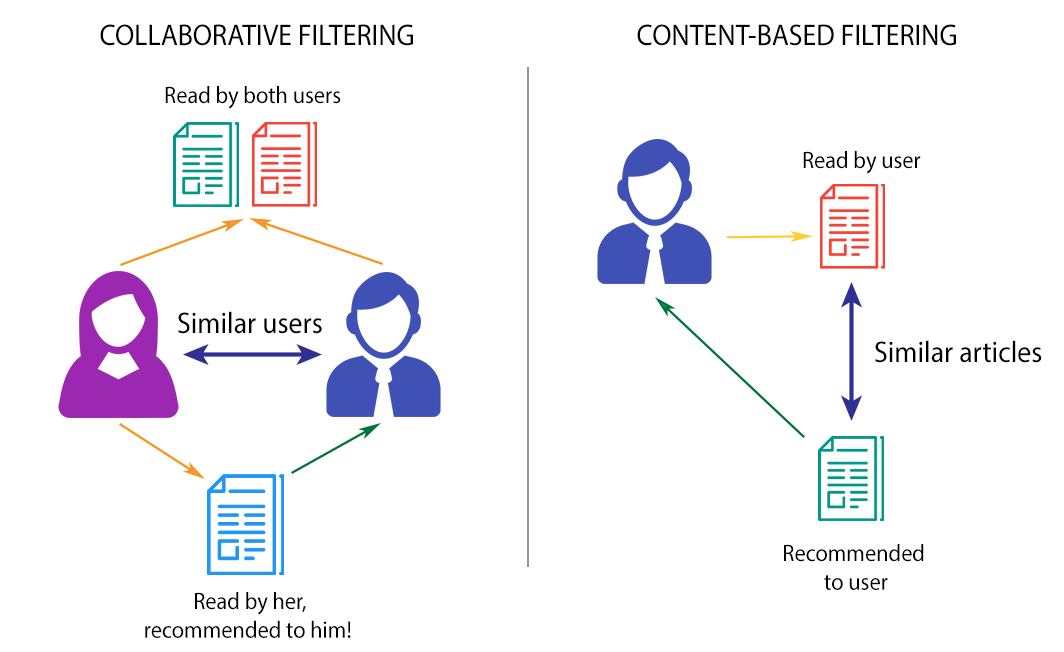
You might have heard the term “Recommendation System (RS)” when YouTubers are discussing the latest tactics to get more views or when you or your friends compare the “Recommended for you” list on Netflix. In a nutshell, recommendation systems recommend things that the people might like based on your own watch history or you and friends watch history as a collective.

WHAT ARE RECOMMENDATION SYSTEM?

A **recommender system**, or a **recommendation system**, is a subclass of [information filtering system](https://en.wikipedia.org/wiki/Information_filtering_system) that seeks to predict the “rating” or “preference” a user would give to an item. They are primarily used in commercial applications.

Examples of such applications include recommending products on Amazon, music on Spotify, and of course, stories on Medium. The famous [The Netflix Prize](https://en.wikipedia.org/wiki/Netflix_Prize) is also a competition in the context of recommendation systems.

denotes a user, i denotes an item, and R is the utility of the user being recommended with the item. Items are then sorted by utility and top N items are presented to user as recommendation.



# **Music recommender system**

One of the most used machine learning algorithms is recommendation systems. A **recommender** (or recommendation) **system** (or engine) is a filtering system which aim is to predict a rating or preference a user would give to an item, eg. a film, a product, a song, etc.

Which type of recommender can we have?

There are two main types of recommender systems:

* Content-based filters
* Collaborative filters

Content-based filters predicts what a user likes based on what that particular user has liked in the past. 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

Recommendations done using content-based recommenders can be seen as a user-specific classification problem. This classifier learns the user's likes and dislikes from the features of the song.

The most straightforward approach is **keyword matching**.

In a few words, the idea behind is to extract meaningful keywords present in a song description a user likes, search for the keywords in other song descriptions to estimate similarities among them, and based on that, recommend those songs to the user.

How is this performed?

In our case, because we are working with text and words, **Term Frequency-Inverse Document Frequency (TF-IDF)** can be used for this matching process.

### 2) Collabarative filtering

Collaborative filtering is the predictive process behind [recommendation engines](https://www.techtarget.com/whatis/definition/recommendation-engine). Recommendation engines analyze information about users with similar tastes to assess the probability that a target individual will enjoy something, such as a video, a book or a product. Collaborative filtering is also known as social filtering.

Collaborative filtering uses [algorithms](https://www.techtarget.com/whatis/definition/algorithm) to filter data from user reviews to make [personalized](https://www.techtarget.com/searchcustomerexperience/definition/personalization) recommendations for users with similar preferences. Collaborative filtering is also used to select content and advertising for individuals on social media.

Three types of collaborative filtering commonly used in recommendation systems are neighbor-based, item-to-item and classification- based.

In neighbor-based filtering, users are selected for their similarity to the active user. This similarity is determined by matching users who have posted similar reviews. Based on the previous similarity, it is presumed that future likes and dislikes will also be similar. From the average rating of the group, recommendations are made for the active user.

An item-to-item filtering process uses a matrix to determine the likeness of pairs of items. Item-to-item processes then compare the current user’s preference to the items in the matrix for similarities upon which to base recommendations.

 A classification-based collaborative filtering system recommends things based on how similar users liked that classification or genre. It is assumed that users that enjoy or dislike similar experiences within a classification will also enjoy others within that classification.

**METHDOLOGY**

So, a content-based recommendation algorithm has to perform the following two steps.

First, extract features out of the content of the song descriptions to create an object representation.

Second, define a similarity function among these object representations which mimics what human understands as an item-item similarity.

How is this performed, then? — you might be wondering —

In our case, because we are working with text and words, **Term Frequency-Inverse Document Frequency (TF-IDF)** can be used for this matching process.

**TF-IDF** is a technique used for information retrieval. It weights a term’s frequency (TF) and its inverse document frequency (IDF); two concepts that we are going to explain later.

The algorithm finds the score for TF and IDF for each term in the document.

After that, the product of TF and IDF of each word is obtained. This is called the TF-IDF weight of that term.

When we use this technique, we are just counting the occurrence of each keyword in a document, and finding its importance by calculating the TF-IDF score for that document.

The higher the TF\*IDF score, the more strange is that word in that context and thus, the more important the term.

Now, let’s explain what term frequency means.

The **Term frequency** of a word in the current document is solely the number of times that it appears to the total number of words in a document.

For example, for the word music in the document I love music because it makes me feel like I can fly

TF(music) = Number of times music appears/Total number of words = 1/12

Simple, right? But what about inverse document frequency?

**Inverse document frequency** of a termis the measure of how significant that term is in the whole corpus.

It is defined as the total number of documents in a corpus to the frequency occurrence of documents that do contain the term following the formula:

IDF = log(Total number of documents/Number of documents containing the term)

If a word is very rare, it means that it has fewer occurrences, and as a consequence, the IDF increases.

Because the TF-IDF score is used to evaluate how important a word is to a document in a corpus, the importance of that word increases when the number of occurrences increases but it is offset by the frequency of the word in the corpus.

If we calculate the TF-IDF score for each word, we have a vector that is commonly called the **TF-IDF Vector**.

Great! But How do we translate this into Python code?

So imagine that we have the [following dataset](https://www.kaggle.com/mousehead/songlyrics/data).

This dataset contains name, artist, and lyrics for 57650 songs in English. The data has been acquired from LyricsFreak through scraping.

We want to build a content-based recommendation system using the TF-IDF technique.

Likewise, we are going to use TfidfVectorizer from the Scikit-learn package.

And as always, pandas will help us read the dataset into a DataFrame.

We can notice also the presence of \n in the text, so we are going to remove it.

After that, we use TF-IDF vectorizer that calculates the TF-IDF score for each song lyric, word-by-word.

Here, we pay particular attention to the arguments we can specify:

* analyzer: Whether the feature should be made of word or character n-grams.
* stop\_word: Remember that stop words are simply words that add no significant value to our system, so they should be ignored by the system. We pass English so that it is identified as the language of the lyrics.

We create a lyric\_matrix variable where we store the matrix containing each word and its TF-IDF score with regard to each song lyric.

But now, the following question arises:

— How do we use this matrix for a recommendation? —

The answer is a word: similarity. We now need to calculate the similarity of one lyric to another.

How do we do that?

For this aim, we can use different metrics such as [cosine similarity, or Euclidean distance](https://towardsdatascience.com/how-to-measure-distances-in-machine-learning-13a396aa34ce), among others.

For our song recommendation system, we are going to use [**cosine similarity**](https://towardsdatascience.com/how-to-measure-distances-in-machine-learning-13a396aa34ce#06cb)and particularly, its [implementation](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html) from Scikit-learn.

**Cosine similarity** is a metric, helpful in determining, how similar the data objects are irrespective of their size. We can measure the [similarity between two sentences in Python](https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/) using Cosine Similarity. In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine similarity between two vectors is

Cos(x, y) = x . y / ||x|| \* ||y||

where,

* **x . y** = product (dot) of the vectors ‘x’ and ‘y’.
* **||x||** and **||y||** = length of the two vectors ‘x’ and ‘y’.
* **||x|| \* ||y||** = cross product of the two vectors ‘x’ and ‘y’.

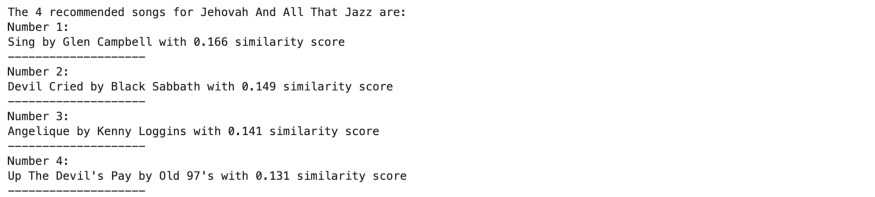
We want to calculate the cosine similarity of each item with every other item in the dataset. So we just pass the lyrics\_matrix as argument.

Once we get the similarities, we’ll store in a dictionary called similarities, the names of the 50 most similar songs for each song in our dataset.

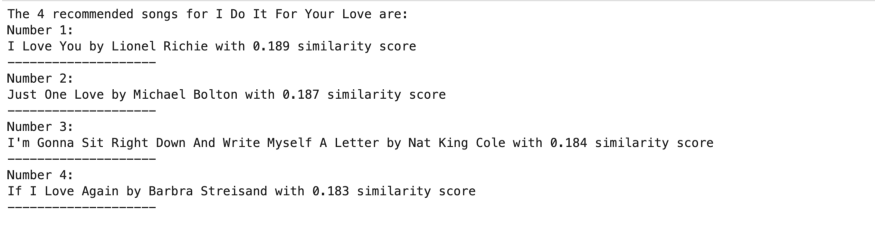
After that, all the magic happens. We can use that similarity scores to access the most similar items and give a recommendation.

First, we’ll define our Content-based recommender class.

And, now we are ready to pick a song from the dataset and make a recommendation.



And we can pick another random song and recommend again:



RESULT

So , hence we have now made the music recommendation system using the basis of cosine similarity and TF-IDF.

**BIBLOGRAPHY**

* [**https://www.geeksforgeeks.org/collaborative-filtering-ml/**](https://www.geeksforgeeks.org/collaborative-filtering-ml/)
* [**https://developers.google.com/machine-learning/recommendation/content-based/basics**](https://developers.google.com/machine-learning/recommendation/content-based/basics)
* [**https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022**](https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022)