
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2022 SCPE. Volume 23, Issues 3, pp. 103–114, DOI 10.12694/scpe.v23i3.2005

MUSIC INFORMATION RETRIEVAL USING SIMILARITY BASED RELEVANCE RANKING TECHNIQUES

KARTHIK VASU* AND SAVITA CHOUDHARY†

Abstract. The purpose of this proposed study activity is to construct a system for the job of automatically assessing the relevance of music datasets, which will be used in future work. Determine item similarity is an important job in a recommender system since it determines if two items are similar. Participants' systems must provide a list of suggested music that may be added to a given playlist based on a set of playlist characteristics, which will work along with the algorithms designed to provide other similar songs. Specifically, in this study, the challenges of detecting music similarity only on the basis of song information and tags given by users have been addressed. The proposed technique has been tested using a variety of machine learning algorithms to see how well it performs. tf-idf and Word2Vec are the methods used to model the dataset and generate feature vectors. It has also been found we that machine learning techniques, including Collaborative Filtering, KNN, Frequent Pattern Growth, and Matrix Factorization, have a greater influence on relevance ranking than traditional methods.

Key words: Music Information Retrieval, Machine Learning, Collaborative Filtering, Spotify

AMS subject classifications. 68P20, 68T05

1. Introduction and examples.

It is the search and organization of enormous collection of music, or musical information, as per their relevance to particular queries that is the subject of Music Information Retrieval (MIR). This is especially significant in light of the large amounts of musical information that is now accessible in digital format, as well as the widespread use of music-related digital services. Aside from that, given its apparent commercial appeal, the majority of media content owners as well as distributors (e.g., Philips and Sony), as well as major technology companies (e.g., Apple and IBM), are actively engaged in research in the area, and a growing number of libraries are attempting to integrate some form of support for MIR into their on-line digital services. This results in an analysis of the text against the text data connected with album and& songs, making the system practically identical to any text-based search engine in terms of functionality (e.g. Google, Yahoo). Although this is the case, systems that are capable of receiving "musical" enquiries, like musical scores, whistling melodies (query by humming), or audio recording segments are necessary due to the nature of the content being retrieved.

The phrase "song similarity" refers to the measurement of how close two songs are close in terms of how probable it is that user really want listening to them when they are compared side by side. Yes, the process of developing an objective similarity measure is subjective, and researchers have taken two approaches to accomplish this: Both the objective technique, in which likeness is identified related to the raw data, such as spectral or rhythmic analysis of songs, and the subjective approach, wherein user-generated data, such as tags (also known as collaborative filtering), are used, are discussed further. For the sake of this research, we shall use the subjective technique to determine the extent to which two tracks are comparable to one another. We would then create a resemblance level among two songs scale from 0 (totally distinct) to one (identical), and then we will determine it using the co-occurrences of pairs of things in users' histories by using cosine metric to determine how similar two songs are to one another. For more information, please see the following link. This measure will also serve as our model of reality and, as a result, as our source of truth in the future. Such a concept is realistic, as shown by the fact that researchers in the area have utilized it successfully [1].

*Assistant Professor, Department of Information Science and Engineering, M S Ramaiah Institute of Technology, Bengaluru, India (karthikv@msrit.edu).

†Associate Professor, Department of Computer Science and Engineering, Sir M. Visvesvaraya Institute of Technology, Bengaluru, India (savitha_cs@sirmvit.edu)

The intended objective of this work is to employ Data Science methodologies in conjunction with Machine Learning algorithms to estimate song popularity on Spotify as well as other music streaming services, which are quite popular and extensively used. Using success measures or criteria, we will evaluate each and every popular Machine Learning algorithm and choose the best algorithm from among those evaluated. It is often the result of some form of calculated miscalculation. The purpose of the best model that has been built is to forecast the popularity of a song based on a variety of current and historical characteristics.

2. Related Work. Music data retrieval, recommendations, and similarity approaches will be discussed in this session. Recommendation system and Relevancy rankings, music information retrieval research, and the tradeoff among short- and long-term involvement in digital sites are all areas of study that we draw upon in our work [2].

2.1. Spotify. The fact that Spotify is in business after three of the world's largest corporations, Google, Apple and Amazon have all entered the music streaming sector is testament to the company's tenacity. In comparison to the other three, they are able to provide superior music recommendations to the user. At first glance, it seems like Spotify has a far superior system than the other three competitors. Spotify recommends music based on three different types of models [3]. The first of them are models of Natural Language Processing. These compare songs based on the words used to describe the music, which may be found in things like articles on the internet. The following is used to propose songs that aren't as popular as the previous. These Content-Based models analyse the real audio and utilise commonalities to suggest comparable songs to you based on your preferences. Collaborative filtering is the name given to the final style. This is accomplished by establishing a user vector for each individual and a song vector for each individual song in the music library [4]. It then compares them in order to propose music that is comparable to each other as well as music that similar people like listening to [4]. As discussed in this article, Collaborative Filtering is the most widely used technique of music recommendation on Spotify and is at the core of Spotify's most popular music suggestion method: Discover Weekly. It will be this algorithm, and more precisely how linear algebra is employed to power it, that will be the focus of our discussion. Notably, other firms, such Last.fm, have taken a similar approach to Spotify's strategy, while another has been promoted by the Netflix Prize, which is modelled after Spotify's concept.

2.2. Collaborative Filtering. Filtering in a Collaborative Way Every week, Spotify creates a Discover Weekly playlist for each of its 140 million customers [5]. Using over 40 million tracks, they discover the songs that are most likely to be a good fit for you and your music collection. If your buddy tells you about a specific music x because they found out that you enjoy song y, Collaborative Filtering aims to mimic that. For this, a latent factor model, a machine learning method, is utilized, which converts unobservable raw data into unobservable latent features. Cost functions are used to calculate the latent components using an alternating least-squares approach. Switching back & forth between user and song factors is the procedure of alternating-least-squares in our example. This approach keeps the cost function converged [6]. A cost function is used to determine how inaccurate an estimate is compared to the true value of the data. In order to solve the problem, you enter a certain set of parameters. Then, using our alternating-least-squares technique, you may fine-tune these characteristics until the cost function converges. When that happens, we'll have a preferences-confidence pair for users to display our choices and the degree to which we're confident in them.

3. Materials and Methods.

3.1. Dataset. The Spotify Million Playlist Dataset (MPD) is a collection of 1,000,000 user-created playlists. From January 2010 until October 2020, these playlists were compiled. Some playlists from original playlists are removed to match the competition's challenge playlist, and all holdout tunes appear in the MPD Test Set 1000 playlists. The initial set of test sets will be the train set that removes rails. When addressing this issue, we refer to "item" as a "song" and "user" as a "playlist". We won't utilise content-based filtering since the dataset doesn't give much information about every song. Our emphasis would be on KNN, Collaborative Filtering and Frequent Pattern Growth as well as matrix factorization [7].

Each of the 1,000 slice files in the Million Playlist Dataset makes up one Million Playlist.

mpd.slice.STARTING PLAYLIST ID - ENDING PLAYLIST ID.json is the name pattern for these files.

Table 3.1: Playlists Field

S.NO	Parameter	Outcome
1.	Playlist	1000000
2.	Tracks	66346428
3.	Unique tracks	2262292
4.	Unique albums	734684
5.	Unique Titles	92944

File mpd.slice.0-999.json contains the MPD's first 1,000 playlists, for example mpd.slice.999000-999999.json contains the latest 1,000 playlists.

Info Field: In the info box, you may get basic information about the slice you're looking at: Slicing is based on the number of slices in a certain file. For example, slices 0-999 indicate the file's version, whereas slices generated on and MPD version 1 indicate whenever the slice was produced. The information and playlists fields in each slice file are both JSON dictionaries.

Playlists Field: Typically, this is a collection of 1,000 playlists. An individual playlist is nothing more than a dictionary, with the following entries:

1. The MPD ID of all this playlist is the pid (integer) ranging from zero to ninety-nine,999.
2. name - string - the playlist's title.
3. The playlist's description, if any (this is an optional string). For the time being, the majority of Spotify's playlists do not include descriptions contributed by its listeners.
4. This playlist's time has just recently been updated in the seconds ever since epoch. Whenever the playlist is reloaded, the timings are calculated based on the current day and time in Greenwich Mean Time (GMT).
5. A playlist's total number of distinct artists may be seen in the num artists field.
6. num albums - the playlist's total number of distinct albums.
7. In the playlist, the number of tracks can be found.
8. num followers - how many followers this playlist had when the MPD was generated. This number does not include the playlist creator's followers.
9. This is the number of distinct editing sessions that have occurred. During a two-hour editing window, any tracks that have been added are considered to be part of the same editing session.
10. A playlist's overall length is represented by duration ms (in milliseconds)
11. A collaborative playlist is one in which all members of the group are actively participating. It is possible for many people to submit tunes to a playlist that is shared.
12. A playlist's "tracks" section contains a slew of data about the songs. This array has a dictionary with following fields for each of its elements:

3.2. Recommender. Products like books, movies, music, and friends would be recommended to customers using a recommender system. Recommenders may fall into one of two categories. Content-based filtering and collaborative filtering are two types of filtering. Content-based systems, on the whole, are easier to use but provide less interesting suggestions. Although collaborative systems may be cumbersome to operate and need a lot of user-generated data, they are the most advanced technology available today [8].

3.2.1. Using content as a filtering mechanism. Products may be manually investigated and labelled by experts or consumers according to several categories or qualities. The more characteristics we have, the easier it is to compute the similarity of one thing to other objects and to find related products.

3.2.2. Collaborative filtering. Filtering in groups: a process in which many people work together to improve the quality of the input. Collaborative filtering, unlike content-based filtering, does not need manual la-

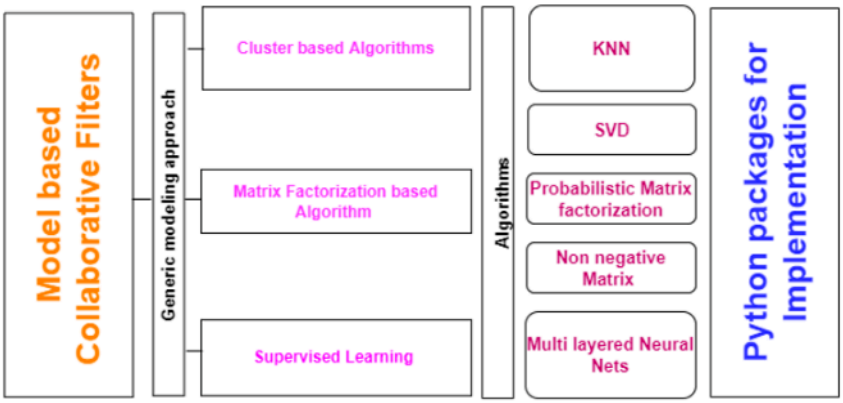

Fig. 3.1: Architecture of Model based Collaborative filtering

belling. To produce product recommendations, the algorithm would seek for other users with similar preferences and habits. There are two forms of collaborative filtering: memory-based and model-based. Clustering and matrix factorization are two machine learning methods that are used in model-based CF in order to determine similarity between user items and their corresponding models.

There are two forms of memory-based filtering: user-item and item-item. Take a certain individual, look for others with comparable ratings to that person, and propose products those people enjoyed. For example, if you're looking for a certain item, you may use item-item filtering to look for additional goods that individuals who like that item also like. Recommendations are generated from things that are entered into the system. "Customers who purchased this item also purchased. It's simple "Customers who are identical to you also enjoyed." and "Customers who like this item also liked..." are the same thing when it comes to Customer Feedback (CF).

The implementation of memory-based algorithms is simple, and the quality of the predictions they make is acceptable. In real-world applications, memory-based CF has limitations since it cannot handle the well-known cold start issue, which arises when new users or items join the system. In contrast to memory-based models, model-based CFapproaches are scalable and therefore can handle greater sparsity levels, but suffers when new members or things that do not have ratings join the system.

Matric factorization is the foundation of all model-based filtering and model-based CF. There are several applications where matrix factorization (CF) is preferred over Memory-based CF, such as recommendation systems.

Users' preferences and the properties of objects may be learned from past ratings using Matrix factorization. Then, using the dot product of these latent features, Matrix factorization can forecast ratings for new items. If you have a user-item matrix, factorization may reorganise it into a low-rank structure that can be expressed as the product of the two low-rank matrices, each of which contains the latent vector. In order to get as close as possible to your original matrix, multiply the low-rank matrix together, which completes the original matrix's missing members.

3.3. Playlist Recommender. The terms "user" as "playlist" and "item" as "song" will be referred for the sake of this discussion. Since the data will not really contain enough detail about every song, we can't utilise content-based filtering. As a consequence, we'd only be able to focus on one thing. KNN, Collaborative Filtering, Word2Vec., Frequent Pattern Growth, ID3, Customized kernel SVM [16].

3.3.1. Collaborative Filtering.

Playlist-based CF: The proposed model will infer the current "rating" from the similarities between each playlist or how other playlists "rate" (include or exclude) a tune. Song-based CF: The model will also estimate the current "rating" from the similarity of each song and how the current playlist "rates" other

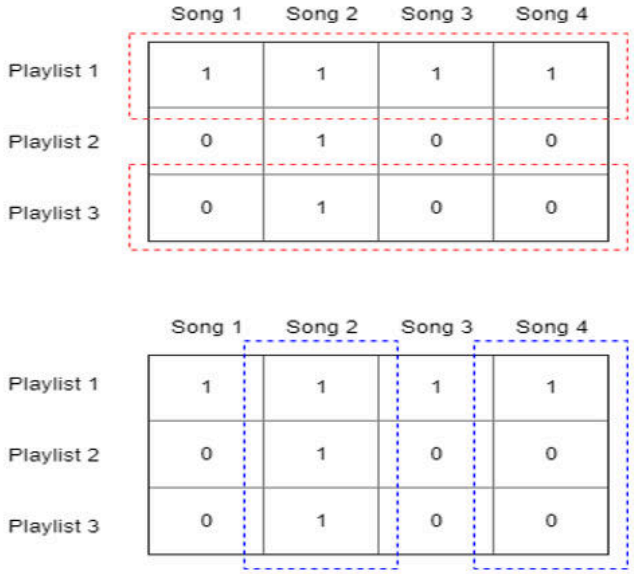

Fig. 3.2: Similarity between song-song or playlist-playlist

songs. This approach has been followed in Collaborative Filtering [17].

Song-based CF: I can estimate the current "rating" from the similarity of each song and how the current playlist "rates" other songs. We followed the approach in Collaborative Filtering.

Step1. Create a playlist-song matrix in which "1" indicates that the song is in the playlist & "0" indicates that it is not. For instance, playlist 1 includes songs 2 and 3, and song 2 is also included in playlist 2.

Step2. To begin, the model divides the data into testing and training sets.

Step3. Determine the degree of similarity among songs or playlists. In playlist-playlist similarities, every row is treated as a vector, but in song-song similarity, each column is treated as a vector.

Cosine Similarity formula is given below in equation 3.1:

$$similarity = \cos(\theta) = \frac{A.B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.1}$$

A_i and B_i are components of vector A and B. We make predictions on the testing set based on similarity matrix and. According to our algorithm, we can predict that song S will be included in any given playlist p by calculating a total of the weighted sums of all other playlists that include the same song. After that, the data is normalised.

$$\hat{r}_{ps} = \frac{\sum_{p'} sim(p, p') r_{p's}}{\sum_{p'} |sim(p, p')|} \tag{3.2}$$

Song-song tries to replace the similarity matrix of playlist with that of songs.

$$\hat{r}_{ps} = \frac{\sum_{s'} sim(s, s') r_{ps'}}{\sum_{s'} |sim(s, s')|} \tag{3.3}$$

Table 3.2: Singular Value Decomposition Equation: **X, U, S, V**^{T}

$\mathbf{X}_{11}$	$\mathbf{X}_{12}$	**...**	$\mathbf{X}_{1M}$		$\mathbf{U}_{11}$	$\mathbf{U}_{12}$	**...**	$\mathbf{U}_{1R}$		$\mathbf{S}_{11}$	**0**	**...**	**0**		$\mathbf{V}_{11}$	$\mathbf{V}_{12}$	**...**	$\mathbf{V}_{1N}$
$\mathbf{X}_{21}$					X_{21}					0	S_{22}				V_{21}			
.				=	.					.					.			
.					.					.					.			
.					.					.					.			
.					.					.					.			
.					.					.					.			
											.	.						
$\mathbf{X}_{M1}$			X_{MN}		U_{M1}			U_{Mr}		0			S_r		V_{M1}			V_{MN}

Table 3.3: Feature Vector

S.NO	Parameter	Outcome
1.	Playlist	1000000
2.	Tracks	66346428
3.	Unique tracks	2262292
4.	Unique albums	734684
5.	Unique Titles	92944

3.3.2. K Nearest Neighbor. Playlist-based (like user-based) :

1. Create a similarity matrix for playlists (cosine, euclidean, Pearson correlation)
2. Px for each playlist
3. 1 = n
4. While total track is less than 500
5. Find the n-th most relevant Px playlist, termed Pr
6. Move K (or all) songs from Pr to Px.
7. Increase n by one.

Song-centered (like item-based) :

1. (The user space is the space of similarity)
2. Create a song similarity matrix (cosine, euclidean, Pearson Correlation)
3. For each playlist, Px:
4. Determine the "cluster centre" by averaging all matches.
5. Obtain K = 500 nearest neighbours and add them to existing songs.

3.3.3. Model Based. Probabilistic factorization (PMF), Orthogonal factorization (SVD) or non-negative factorization are all methods for matrix factorization (NMF). Collaborative Filtering via Singular Value Decomposition (SVD) could be defined as the process of estimating a matrix X through svd. The team that wins in the Netflix Award contest generated product suggestions using SVD matrix factorization models. The generic equation is as follows: $\mathbf{X = U \times S \times V}^T$. The transpose is shown in Table 3.2.

Now that we hide feature vectors for playlists and song, we could inject them into any Machine Learning algorithms for result prediction (Table 3.3).

Figure 3.3 shows the distribution of Playlist length, Number of Albums / Playlist, Number of Artist / Playlist, Number of edits / Playlist, Number of Followers / Playlist, Number of Tracks / Playlist. Song based relevance ranking is described in the figure 3.5. Top 20 songs are listed as per the similarity check using machine learning technique. This describes the song based and artists based relevance ranking. could improve the success of automatic ranking. Based on this intuition, a new method has been introduced for the selection

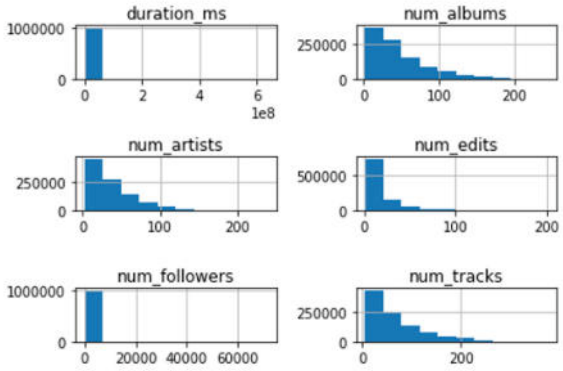

Fig. 3.3: Distribution of Playlists

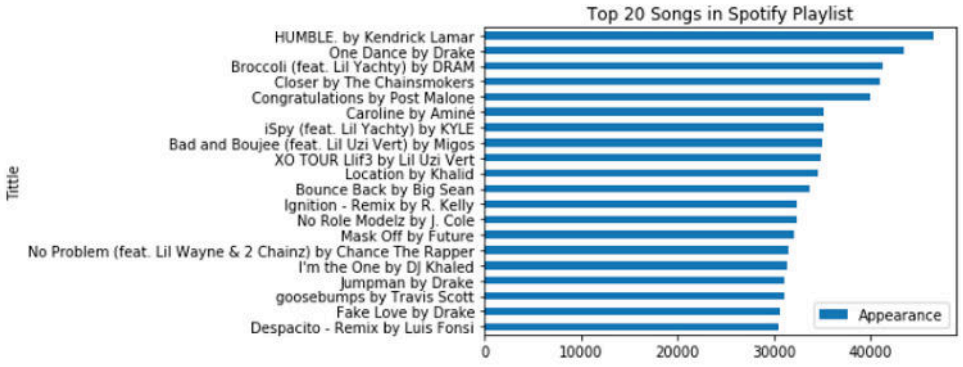

Fig. 3.4: Song Based Relevance ranking

of systems to be used for data fusion. For this purpose, bias concept is utilized that measures the deviation of a system from the norm or majority and employ the systems with higher bias in the data fusion process. This approach provides even higher correlations with the human based results. This demonstrates that the proposed solution outperforms the proposed automatic ranking methods shown in figures 3.4 and 3.5 [20,21,23].

The figures 3.4 and 3.5 show the relevance ranking of songs and artists respectively. The list is of the year 2020. For the search of Top 20 Songs, the list obtained has used the similarity based relevance ranking technique, which is discussed in Section 3. The same applies to the search of Top 20 Artists in the playlist available.

4. Results and Discussions.

4.1. Metrics. The following metrics will be used to evaluate submissions. In other words, all metrics would be analysed at both the level of tracks (the identical track should be replicated) and also at the artist levels (any track by that artist is a match). G's "ground truth" set of music and R's "ordered lists of recommended tunes" has been employed in the following discussion. From: to-subscripts are used in order to index a list, which is signified by ||. On individual metrics, previous entries are scored higher when there are multiple submissions

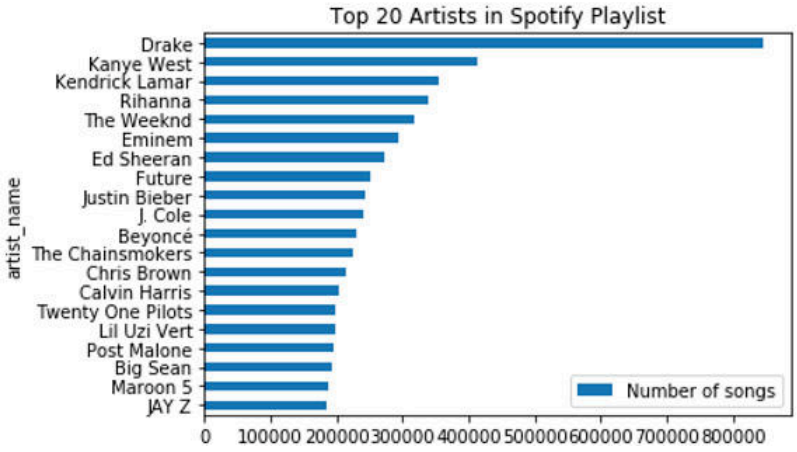

Fig. 3.5: Artists Based Relevance ranking

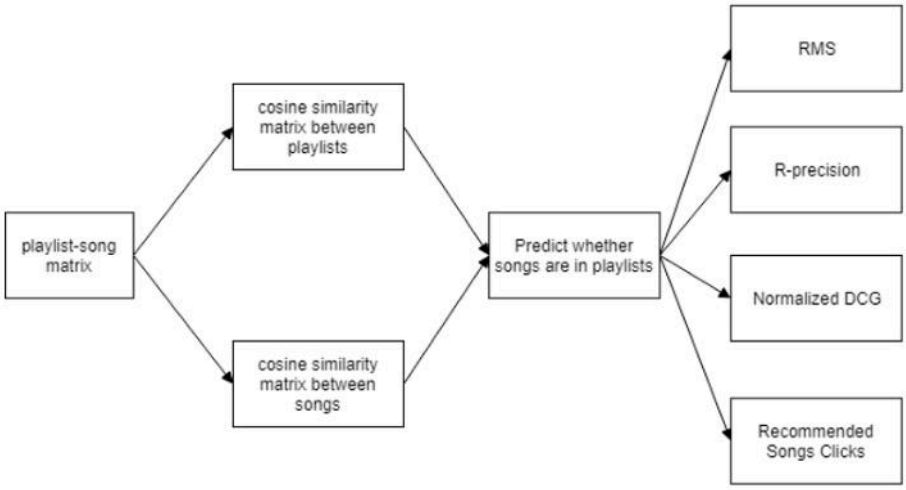

Fig. 4.1: Sample: Similarity between song-song or playlist-playlist

that are tied. Figure 4.1 shows the Similarity between song-song or playlist-playlist [20,21].

4.2. R-precision. It is defined as the number. of essential tracks that were found divided by number of tracks that were already known to be relevant (i.e., the number of tracks that were not found).

$$R - precision = \frac{|G \cap R_{1:|G|}|}{|G|} \tag{4.1}$$

To get this metric, you need to look at all playlists inside the challenge set. This metric is about how many tracks were found that were relevant (regardless of order).

5. Normalized discounted cumulative gain. Recommendation tracks with a higher DCG ranking are more likely to be of interest to users. When calculating the DCG, divide by the optimal DCG wherein the

Table 7.1: Comparison of Performance with Machine Learning algorithms vs Customized Model

Algorithms	Variable	Accuracy/ R-precision	NDCG	Duration
KNN	Song-features	87.04	81.44	36.25
FP growth	Song-features	81.25	76.42	41.28
Support Vector Machine	Song-features	78.23	74..21	57.11
ID3- Iterative Dichotomiser 3	Song-features	76.17	81.77	49.23
Navie Bayes	Song-features	74.20	79.45	44.18
Customized kernel based Support Vector Machine	Song-features	89.21	84.25	31.56

suggested tracks are ideally ranked:

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{\log_2(i+1)} \tag{5.1}$$

In this situation the ideal IDCG & DCG are :

$$IDOG = 1 + \sum_{i=2}^{|G \cap R|} \frac{1}{\log_2(i+1)} \tag{5.2}$$

The DCG is zero if the intersection of R & G is empty. The following formula is used to compute NDCG:

$$NDCG = \frac{DCG}{IDCG} \tag{5.3}$$

6. Recommended Songs clicks. Using Spotify's Suggested Songs function, you can get a list of 10 songs to add to an existing playlist based on the music in it. Ten more tracks could be generated by refreshing the list. The quantity of refresh required before the suitable track is found is called "Recommended Songs clicks." It's done like this:

$$clicks = \left\lfloor \frac{\arg min_i\{R_i : R_i \in G|\} - 1}{10} \right\rfloor \tag{6.1}$$

A value of 51 (i.e. 1 + the greatest clickthrough conceivable) is chosen if the metric (i.e. the necessary track in R) does not exist.

7. Aggregation of Rank. The Borda Count voting procedure will be used to compute the final rankings. The highest ranked system receives p points for each of the p participant rankings based on NDCG, R-precision &, the next system gets p-1 points, Recommended Songs clicks etc. The player with more overall points wins. Top-down comparative assessment is used to determine which systems have the most first-place position in the case of a tie. The result of item and user based search is shown in Table 7.1 and the outcome of an item-item/ customer- items based search is shown in Table 7.2.

It can be concluded from figures 7.1, 7.2 and 7.3 that the custom model has outperformed the above mentioned algorithms, KNN, FP growth, support vector machine, ID3 and Navie bayes. The Accuracy in Figure 7.1 has been proven to be better than the comparison algorithms by 2.17%, while in Figure 7.2, the time taken for a similar match has reduced by about 13%. The NDCG has also seen a growth about 2.81% which can be discerned from Figure 7.3.

8. Conclusion. We are able to produce suggestions of music that are identical to a music in our playlist by using idea of cosine similarity. As a result, you can use the procedures outlined above to create your own Spotify playlist recommender. Possible consequences of machine learning also include individualized, immersive

Table 7.2: The outcome of an item-item / customer- items based search

Procedure	**Variable**	**Accuracy/ R-precision**	**NDCG**	**Clicked songs**	**Duration**
Playlist-based baseline	Playlist	0.7766	1.601	0	41.42
Song-based baseline	Song	0.7847	0.7975	0	4183
Word2Vec + Song-based	100-200-300 dimension	0.003	0.004	10.35	69.13
Word2Vec + Playlist-based	min_fre = 3, dimension 50	0.0171	0.015	8.086	83.25
Word2Vec + Playlist-based	min_fre = 3, dimension 100	0.019	0.0172	7.805	88.35
Playlist-based CF (get top K rating songs)	Song	0.8045	0.8011	0	approx 12000

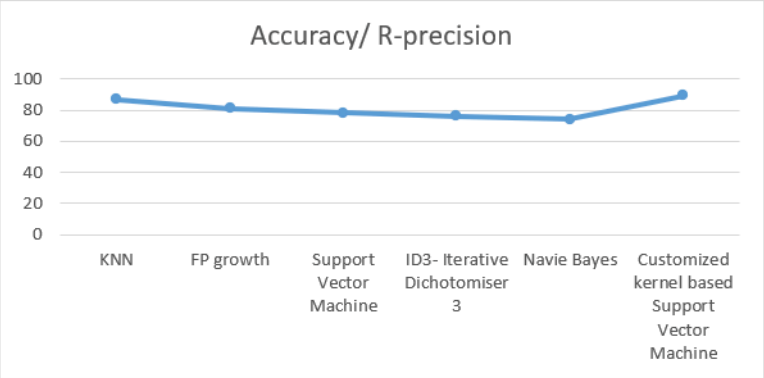

Fig. 7.1: Comparison of Accuracy with Machine Learning algorithms vs Customized Model

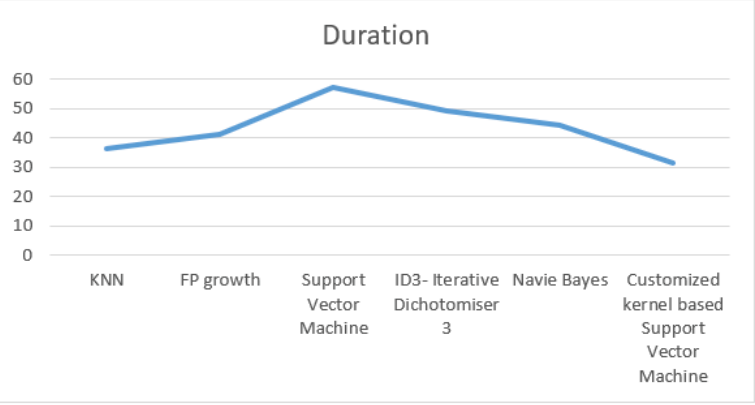

Fig. 7.2: Comparison of Time taken with Machine Learning algorithms vs Customized Model

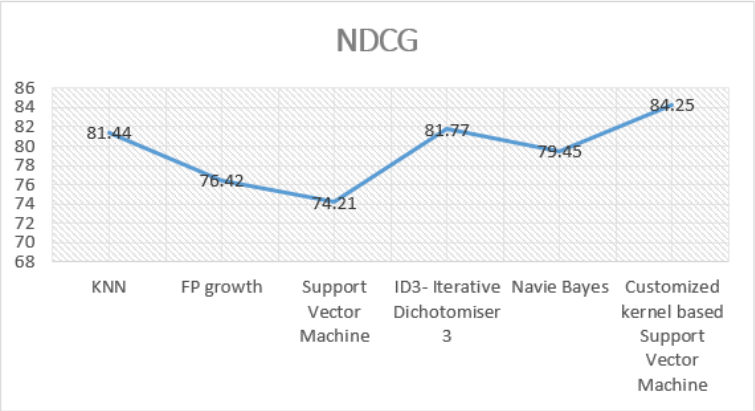

Fig. 7.3: Comparison of NDCG with Machine Learning algorithms vs Customized Model

Table 7.3: RMS and Accuracy Result

Procedures	RMS	Accuracy/R-precision
CF that is playlist-based (used to generate songs having k rating)	0.00716	0.621
Song that is based on CF	0.00014	0.76
Matrix Factorization	0.00045	0.69

guest experiences having increasingly complex features, in parallel with developments in technology and media affordances. Choosing recommendation methods produces better results for the full dataset, however for large datasets, processing techniques are required using big data technologies to fetch the data within the less time which will be handled in future improvements and analyses.

9. Future Scope. With the advent of various similar algorithms, the proposed model has shown promise by being relatively better at obtaining similar songs at the quickest. For huge datasets, the time taken for the searching process can be further reduced using Spark and Hadoop frameworks. There is hope to be a milestone and a reference point for future researches done in the same field as well as for others.

REFERENCES

[1] Cunha, R. L., Caldeira, E., Fujii, L., *Determining Song Similarity via Machine Learning Techniques and Tagging Information.*, arXiv preprint arXiv:1704.03844

[2] Anderson, Ashton, et al., *Algorithmic effects on the diversity of consumption on spotify*, Proceedings of The Web Conference, 2020.

[3] Bateira, José Lage, *Spotify-ed-Music recommendation and discovery in Spotify.*, 2014.

[4] Pichl, Martin, Eva Zangerle, and Günther Specht., *# nowplaying on# spotify: leveraging Spotify information on Twitter for artist recommendations.*, International Conference on Web Engineering, Springer, Cham, 2015.

[5] Ouiem Bchir, Mohamed M. Ben Ismail and Sara Algarni, *Support Kernel Classification: A New Kernel-Based Approach.*, International Journal of Advanced Computer Science and Applications(IJACSA), 11(??), 2020.

[6] Belattar Sara, Abdoun Otman and El khatir Haimoudi, *New Learning Approach for Unsupervised Neural Networks Model with Application to Agriculture Field.*, International Journal of Advanced Computer Science and International Journal of Advanced Computer Science and Applications(IJACSA), 11(5), 2020.

[7] T V Divya and Barnali Gupta Banik, *An Empirical Study on Fake News Detection System using Deep and Machine Learning Ensemble Techniques.*, International Journal of Advanced ComputerScienceandApplications(IJACSA),12(12),2021.

[8] Lan Phuong Phan, Hung Huu Huynh and Hiep Xuan Huynh, *Recommendation using Rule based Implicative Rating Measure.*, International Journal of Advanced Computer Science and Applications(IJACSA),2018.
[9] Kim, J., Urbano, J., Liem, C.C.S. et al, *One deep music representation to rule them all?*, A comparative analysis of different representation learning strategies., Neural Comput & Applic 32, 1067–1093 (2020).
[10] Pampalk, Elias.,, *Computational models of music similarity and their application in music information retrieval.*, Diss. 2006.
[11] Mitra, Bhaskar, and Nick Craswell, *Neural models for information retrieval.*, arXiv preprint arXiv:1705.01509 (2017).
[12] Schedl, Markus, Emilia Gómez Gutiérrez, and Julián Urbano, *Music information retrieval: Recent developments and applications.*,Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261. (2014).
[13] Amudha, S., and I. Elizabeth Shanthi, *Phrase based information retrieval analysis in various search engines using machine learning algorithms.*,Data Management, Analytics and Innovation. Springer, Singapore, 2020. 281-293.
[14] Lin, Jimmy, et al., *Supporting interoperability between open-source search engines with the common index file format.*,Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.
[15] Mitra, Bhaskar, and Nick Craswell, *An introduction to neural information retrieval.*,PBoston, MA: Now Foundations and Trends, 2018.
[16] Schindler, Alexander, *Multi-Modal Music Information Retrieval: Augmenting Audio-Analysis with Visual Computing for Improved Music Video Analysis.*,arXiv preprint arXiv:2002.00251 (2020).
[17] Myna, A. N., K. Deepthi, and Samvrudhi V. Shankar, *Hybrid Recommender System for Music Information Retrieval.*,Journal of Computational and Theoretical Nanoscience 17.9-10 (2020): 4145-4149.
[18] Knees, Peter, Markus Schedl, and Masataka Goto, *Intelligent user interfaces for music discovery.*,Transactions of the International Society for Music Information Retrieval 3.1 (2020).
[19] Castellon, Rodrigo, Chris Donahue, and Percy Liang, *Codified audio language modeling learns useful representations for music information retrieval.*,arXiv preprint arXiv:2107.05677 (2021).
[20] Karthik, V., and Savita Choudhary, *TaCbF-"Trending Architecture for Content based Filtering using Data Mining.*,Proceedings of International Conference on Intelligent Computing, Information and Control Systems. Springer, Singapore, 2021.
[21] Gupta, Rahul, Jayesh Yadav, and Cheshtha Kapoor, *Music information retrieval and intelligent genre classification.*,2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), IEEE, 2017.

Edited by: Vinoth Kumar
Received: May 5, 2022
Accepted: Aug 23, 2022