Music Recommendation System Using Deep Learning and Language Modelling

GAURAV GARG

Thesis Report

MARCH 2022

DEDICATION

My dissertation is dedicated to my family and many friends. Sushil and Rakhi Garg, my beloved parents, whose words of support and drive for tenacity echo in my ears, deserve special thanks. Ayushi, my sister, has never left my side and is one of a kind. This dissertation is also dedicated to my numerous friends who have helped me during the process. I will always be grateful for what they have done for me. I dedicate this work and express my gratitude to my JPMC colleagues and mentor.

ACKNOWLEDGEMENTS

I would like to acknowledge my thesis supervisor Akshita Bhandari for her guidance throughout my thesis work. Having the opportunity to be taught by a subject master has been both a privilege and a delight. Thank you very much for being the kind of supervisor that every student requires. For a young academic, this is the ideal role model.

I would also like to thank my family and friends for encouragement and support over the completion and fulfilment of my research project. I would dedicate my thesis to my parents and sister.

ABSTRACT

This study talks about usage of music recommendation system for the task of Automatic Playlists Completion. The evolution of recommender system is discussed from Information retrieval and collaborative filtering to machine learning and deep learning. The main focus in this study is the sequential nature of playlist and the idea of tracking the sequential pattern for making recommendations. This study uses Spotify's million playlists dataset for the analysis. The experimental results of proposed methods tested on real-world dataset is also included in this study. This study compares the proposed solution with other state-of-the-art approaches and evaluates the results on three different evaluation metrics defined. At the end, study concludes with the findings and steps for future work that could improve the performance of the system.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

APC – Automatic Playlist Continuation

UI – User Interface

RS – Recommendation System

MRS – Music Recommendation System

CF – Collaborative filtering

CBF – Content-based filtering

SVD – Singular value decomposition

AI – Artificial Intelligence

CNN – Convolutional Neural Network

RNN – Recurrent Neural Network

ANN – Artificial Neural Network

DNN - Deep Neural Network

IR - Information Retrieval

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

Recommender systems have become a necessary component of many e-commerce and Internet-based organisations such as Google, YouTube, Facebook, Netflix, LinkedIn, Amazon, and others have used recommender systems as part of their business strategies since the Web's inception and spectacular expansion over the last decade. firms in various domains like songs, movies, articles, shopping, etc. These systems are based on user sentiments about a product or service with the goal of promoting them to other users with similar interest according to their past behaviours and tastes. YouTube, for example, offers recommendations for a user based on their viewing patterns and searches for like users. The end goal with automatic playlist continuation (APC) is for music recommendation systems to find tracks that fit inside a specified playlist. A big portion of automatic playlist continuation mission is to appropriately deduce the playlist's intentional purpose. It is difficult not only because of the wide range of intentional purposes (assuming they exist at all), but also because of the wide range of corresponding attributes or characteristics that will be required to hypothesize those purposes. A recommender system is an essential and appealing domain in information filtering systems that seeks to forecast active users' choice or rating of a particular item. A recommender system is a sophisticated engine that uses a variety of ways to suggest the most relevant products for a group of users based on their prior actions and preferences.

Therefore, relevant and appropriate recommendations are extremely important for these businesses to attract more users into using their platform for longer period of time. Before the digital revolution, music industry relied on a variety of ingress points into the music catalogue: trending, playlists by mood, genres, what’s new, hits, and created platforms for sharing playlists. In contrast to other creative content, a track is just a three-minute entertainment which requires suggestions more often than other sources of entertainment. As a result, the album's historic format, which ensures a minimum permissible duration, as well as its aesthetic objective, is preserved. Listeners now have devices, apps, and algorithms thanks to the digital revolution, which allow them to capture those listening opportunities and adapt to detailed user interfaces (UI) on PCs, simplified UI on mobile phones and cars, virtual assistant controlled by voice and smart

speakers. Listeners can experience rich navigation with smart earphones today where no screen interaction is required. Higher granularity (the ability to produce numerous types of playlists, similar genre and artist), more frequent selection updates, deeper dive into the collection, and personalisation are all advantages of digitization. Various platforms like Rapcaviar, Discover Weekly provides listeners with customised playlist updated every week. Spotify on the other hand, creates multiple playlists based on users most played artist, popular songs in the area and lots of other playlist categorised by genres.

The collaborative filtering algorithm has been determined to function well based on users' listening behaviour and past ratings. When combined with the use of a content-based model, the user can get a list of songs that are comparable based on low-level acoustic variables like rhythm and pitch, as well as high-level features like genre and instrument. Last.fm2, Allmusic3, Pandora4, and Shazam5 are 4 music discovery services that have successfully used these two methodologies. Meanwhile, these websites give a one-of-a-kind platform for retrieving rich and usable data for user studies. There is a lot of possibility for future expansion thanks to new discoveries in psychology, signal processing, machine learning, and musicology. As a result, this study examines a broad music recommender framework, including user profiling, item modelling, and item-user profile matching, as well as a number of state-of-the-art methodologies. We conclude and suggest a new model based on user motivation at the end of this study.

1.2 Problem Statement

In the last decade, recommender systems have been in the focus for many studies to develop new methods and improve their accuracy. Two types of recommender systems are: collaborative filtering and content-based filtering methods. Most recommender systems are based on collaborative filtering as it is feasible and easy to implement. There are two categories of collaborative filtering: memory-based (Li et al., 2016) and model-based strategies (Paradarami et al., 2017). Model-based methods use ratings to train a model and predict ratings on unrated items by users, whereas memory-based methods use similarities between items and users to predict. Researchers (Zarzour et al., 2018) introduced a model-based music recommendation system which clusters users using k-means clustering and reduce dimensionality with SVD. One of the earliest model-based collaborative filtering methods was proposed 2 decades ago (Chen and George, 1999), which was based on the idea that users (or judges) who tend to give similar ratings can be modelled as having identical rating probabilities.

Content-based methods take into account both the profile of user's interest and description of item. Bayesian classifiers, for example, are used to assess the likelihood of a user loving an item based on its content (Pazzani and Billsus, 2007) and another research (Chen et al., 2008) introduced contest-based recommender systems that employs profitability into CF-based systems. A text similarity method was introduced (Robertson and Walker, 2000) that sets a threshold for text similarity in item description and employs adaptive filtering. These systems suffer from many challenges such as misinterpret songs that rarely appear, completing playlist with very few tracks, neglecting sequence of tracks in a playlist. To address these challenges, there are hybrid methods proposed (Nicholas, 1999) and (Ungar and Foster, 1998) which combines collaborative features and content together. Recently, researchers (Xiong et al., 2018) proposed a deep learning-based hybrid method by combining collaborative filtering and textual content. Furthermore, explainable recommendation has also been a field of research interest (Zarzour et al., 2020) where trustworthiness score which depends on users' reactions after watching recommended videos is combined with collaborative filtering.

These collaborative filtering and content-based methods are also applied in music recommender systems. Some authors (Chen et al., 2012) and (Mcfee and Lanckriet, 2011) have developed ways for modelling the transitions between songs in playlists using Markov chains. Although, recently researchers have found that sequence of songs hardly matters to listeners while the song-to-song transitions (Kamehkhosh et al., 2018) and ensemble of songs in playlist (Chen et al., 2018) seems to matter. An online competition (ACM RecSys Challenge 2018) was held for the task of Automatic playlist continuation (APC), which involves recommending songs that are likely to be added to an existing playlist. For example, the recommended songs for a list of new year songs should be other new year songs. As part of the challenge and to promote this type of research at scale, Spotify publicly released the Million Playlist Dataset (MPD) in 2018. The data is now publicly available at (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021) . Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every track. Playlists are filtered to remove tracks with offensive content.

The methods used by participating team in the challenge were mostly based on collaborative filtering combined with dimensionality reduction or CNN models. For instance, (Yang et al., 2018) proposed a 2-phase collaborative filtering model where in first phase autoencoders is used to encode and decode playlist features and CNN is used for training just playlist titles. Another team (Teinemaa et al., n.d.) used a three-stage model, where in first stage 4 collaborative filtering models are built on 4 context track-track, word-track, album-track and artist-track. First stage is then followed by taking weighted sum of all 4 models and then completing playlist for a given number of tracks based on same album and popular tracks. The winning team (Volkovs et al., 2018) also applied a two-stage model with additional node for cold start problem. Cold start problem occurs when there are only 2 playlist features available: name & length of the playlist. The first stage employs a WRMF collaborative filtering method which is one of most known methods for implicit/binary collaborative filtering (Hu et al., 2008), since WRMF ignores order of songs in the playlist it is clubbed with a temporal CNN and Neighbour-based CF models (song-song, playlist-playlist). The second stage contains an xgboost classifier used to re-rank prediction from 1st stage and additional playlist features like average song/artist/album, song features like artist/album, pairwise playlist song features like similarity between target song and songs in the playlist and furthermore, creative features like danceability, energy, mode etc. taken from spotify audio api.

With the recent improvement in deep learning techniques, many types of powerful news-based (Okura et al., 2017), apps-based (Cheng et al., 2016) and video-based (Covington and Adams, 2016) recommendation systems are based on deep learning. To address the issue of data sparsity and cold-start recently many deep learning techniques are proposed. Another strong technique, Collaborative Deep Learning (CDL), is given, which uses Bayesian formulation to do deep representation learning for the ratings matrix. CDL allows for two-way communication between content information and ratings matrix feedback.

However, there is no study that demonstrates how to embed people and products together. with taking order of songs into account. Also, there are no studies that combine recent deep learning models with cold start problem to build a versatile collaborative filtering model. For example, by clubbing a neighbour-based collaborative filtering model with deep learning and employing a CNN trained on playlist titles to solve for cold-start problem can produce a more effective result. There are also no studies that take

number of times a song appears in playlist into consideration.

1.3 Research Questions

The aim of this research is to develop a recommender system that performs Automatic Playlist Continuation (APC). Since playlist continuation is required to be done in production environment, the recommender system should be versatile enough to work on various limitations such as a smaller number of tracks in the playlist, only playlist title etc. and hence will be tested on such a test set.

Based on the literature review, below research questions have been formulated:

- What are the most and least important playlist and track features during APC?
- What are the best performing models to recommend songs for playlist?
- What solves cold-start problem for playlist with very less features and tracks?
- What can be laid as a foundation for future work in music recommendation systems?

1.4 Aim and Objectives

The research aims to understand factors influential in music recommendation systems with the task of Automatic Playlist Continuation. The research targets development of an efficient music recommender system to be used by online music streaming websites and eventually help listeners find more tracks they may like.

Below are the research objectives:

- To perform literature review of researches in recommendation systems.
- To perform exploratory data analysis on track and playlist features, find correlation between them and perform feature generation.
- To employ deep learning, clustering and collaborative filtering models for Automatic playlist completion.
- To compare results and identify most efficient models in creating a music recommender system.
- To consolidate results from finalized models.

1.5 Significance of Study

The study is directed to develop a music recommender system that can be used in production by music streaming platforms to give useful and relevant recommendations to its users. This study will focus on developing a solution that combines recent deep learning models with cold start problem to build a versatile collaborative filtering model

which has not been explored yet. The research would also be able to answer which features contribute most and least to music recommendation, this can also greatly help streaming platforms to better understand profile of the user. This study introduces a new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques with the goal of increasing the program's recommendation performance. Music recommendation is relatively a new area of research and is gaining researchers interest over the years since the business of online music streaming platforms entirely depend on their ability to keep user attracted to the platform for as long as possible and at the heart of it sits the recommendation system.

1.6 Scope of the study

Recommendation is all about broadening a listener's musical horizons beyond something that they already know and enjoy. It gives listeners more navigation speed once they've utilized all of their song/artist search abilities. Scope of this study is to make use of the existing deep learning, collaborative filtering and clustering algorithms and to develop new such algorithms that best suits the recommender system. The evaluation of model would be carried out on the separate test set which has playlists of varying lengths, the evaluation metrics used can tell us about recall and top k accuracy of the model. The suggested models will be trained and tested to get consolidated table and arrive at a conclusion on which models performs the best and if there is a scope of further development in the models.

1.7 Structure of the study

The report is structed into below 6 sections:

- The goal of the research, background, problem statement, significance and scope of research is explained in Chapter 1.
- Chapter 2 contains detailed explanation of review conducted to gather information and knowledge from historical research works, how recommender systems have evolved and what challenges are yet to be addressed.
- Chapter 3 talks about the approach followed to conduct the experiments, steps required to process the dataset and explanation of the proposed solution.
- Chapter 4 includes implementation of the approach decided, steps followed for data cleaning, model building and summary.
- Results and Evaluation of experiments conducted are explained in Chapter 5.
- Chapter 6 includes conclusion followed by future work and recommendations.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

With the pool of numerous music and video streaming services, thanks to the recent evolution in web industry, we have access to an unlimited amount of music that too with no cost. Before this evolution, movies were the only platform for artist to release their music specifically in India via Bollywood. As these music streaming services evolve, many underrated artists get to promote their work and are recognised based on it and hence there are a sudden increase in number of tracks available out there. In addition, there are artists who release cover version of an existing track which leads to another problem of finding the right one in presence of different versions of the same track. This trouble of searching for an interesting piece to listen to can be minimized by a good recommendation system that works on users' preferences, genre, what's latest and trending and is able to carry on the mood set by previous tracks in the playlist.

Recommendation systems (RS) have emerged and evolved during the last couple of decades. In the recent years, research on recommendation systems experienced an exponential growth due to competitions like the ACM Recommender Systems Challenge 2018 conducted by Spotify (ACM RecSys Challenge 2018). To calculate individualised suggestions, the majority of music recommender systems (MRSs) are now relying on usage patterns, implicit feedback or explicit evaluations, which are used by collaborative filtering (CF) models. Neighbourhood models are a type of classic CF model that score required user-item recommendation based on item and user neighbourhood, taking into account that they are users with similar behaviour or tracks with similar music. More contemporary RSs, on the other hand, are more likely to utilise model-based CF variations like matrix factorization (MF), which is a parameter-based model whose parameters are learnt by optimization framework like Bayesian customised ranking (BPR) developed by (Cheng et al., 2016).

Despite its rapid rise in popularity, current MRS research on deep learning faces a number of obstacles. First and foremost, as with all DL jobs, transparency is a concern. This pertains to the transparency of why an MRS chooses whether a certain music piece is offered to the target user, i.e., the system's thinking while generating predictions, in the

realm of music recommendation. While some conventional machine learning approaches, like as rule learners or decision trees, can provide reasons for their classification or regression decisions, the latent variables or embeddings utilised in today's DL-based MRS do not. One approach to addressing this issue is to combine classic supervised techniques with DNN, as advocated by Zhang et al (Xiong et al., 2018).

2.2 Recommender Systems

Recommender systems have a widely used in music/video streaming platforms, e-commerce, application stores, gaming websites, etc. Their main goal is to generate meaningful suggestion personalized for the user. They offer various advantages to the companies like increase user spending, diversify user choices, increase user dependency and satisfaction, understand the user-item behaviour. There are several approaches followed in the last 2 decades to develop RSs based on the use case, this study talks about these approaches in detail.

2.2.1 Evolution

With the exception of Shardanand, research on music recommendation did not begin in earnest until 2001 (Celma 2010). (1994). Academics and industry professionals have worked on music recommendation throughout the previous decade. Search and filtering, musicology, data mining, machine learning, personalization, social networks, text processing, complex networks, user interaction, information visualisation, and signal processing are all topics of research in music recommendation (Celma and Lamere 2011). Collaboration filtering (Shardanand 1994) or content-based filtering are used in traditional music recommender systems (Logan 2004). Collaborative filtering examines previous interactions between users and systems and makes suggestions to the user based on a group of like-minded users' ideas.

In the e-Commerce business, recommendation algorithms play a significant role. Amazon's personalised suggestion lists, personalised Internet radio (e.g., Last.fm and Pandora), and software apps (e.g., Microsoft's Zune Mixview, Apple's iTunes Genius) are all recognised MRSs. Echo Nest13 and BMAT14 are two companies that provide unique solutions that help users find, customise, and filter massive volumes of audio information. The evaluation measure root mean square error (RMSE) has been commonly used to assess recommender systems; however, RMSE is not very effective

when assessing music recommendation systems (Celma and Lamere 2011). Users' demand for diverse and surprising music is ignored when there is a blatant focus on correctness. Furthermore, accuracy measures such as RMSE can only be calculated on test data that has already been assessed by users, which might contribute to skewness in the results. More relevant evaluation metrics will be discussed in this thesis that evaluate a music recommender system in a better way.

2.2.2 Applications

Recommendation system can be advantageous to almost any business. To benefit from RSs there are 2 major points to be fulfilled which are

- Diversity in user behaviour – if there are only few examples of user behaviour to learn from, humans can perform better than RSs
- Sufficient information about users – High number of features available for every user is required to generalise deep insights about users.

According to above two points, following industries gain from RSs

I. Retail – Supermarket data about products bought together is a direct insight about users' preferences and intent. Markets like Walmart, D-mart use this data to perform data mining operations and determine which items generate high utility.
II. Banking – Knowing detailed financial situation of a customer along with their demographics, transactions, etc. can be used to recommend financial products.
III. E-commerce – Similar to retail stores, e-commerce platform uses past purchases to determine user intent. In addition, they can also track surfing information to know which type of product customers are looking for.
IV. Media – Streaming websites use past streaming details and demographics to understand user mood and recommend music/videos accordingly.
V. Telecom – Telecom industries track every customer interaction and know user behaviour throughout the day. They can recommend plans personalized to users according to their usage.

2.3 Collaborative Filtering

Collaborative filtering is a set of strategies used in recommender systems to recommend products to consumers who have previously rated them favourably by other users with similar tastes. The underlying premise is that if two individuals have similar opinions on one item, they are more likely to have similar opinions on other items than two users selected at random.

2.3.1 User-item representation

To utilise this method, you'll need three things: a set of users U, a set of available objects I, and historical data on each user's interactions with specific products. The user-item matrix is the most frequent means of representing data for collaborative filtering. Generally, each row contains one user's interactions, whereas each column contains all users' interactions with a single item. This may be expressed as a matrix R, where the value of Ri,j reflects the user i $\in$ U preference for item j $\in$ I. This matrix's values are generally numerical representations of item user ratings. Table 2.1 shows an example of such a matrix. There is sparsity in this matrix since it has many missing values, which is the usual case in real world scenario for user-item matrix.

Table 2.1 User-item representation matrix

	Track 1	Track 2	Track 3	Track 4
User 1	-	1	-	-
User 2	-	-	1	1
User 3	1	1	-	-
User 4	-	-	-	1

2.3.2 User-based approach

Users with similar behaviour are grouped together and Ri,j values are determined by matrix values of users in the same group. For example, in the given table above, user 2 and 4 will be grouped together as their preference for track 4 matches. Value R (4,3), i.e., preference of user 4 for track 3 will be calculated using user 2's preference. Since both user 1 and 3 have no connection to track 3 and 4, they are not included in this group. This example only has 4 users and 4 tracks, the complexity of grouping the users increases with increasing number of users and tracks.

2.3.3 Item-based approach

Similar to the first approach, here items (tracks) are grouped together to score unknown user-item interaction. Similar items are those that get rated by all users in the same way.

Taking above table as an example, tracks 1 and 2 are grouped together as they have similar ratings by 3 and are not connected to users 2 and 4. Score for R (1,1) will be found based on user 1's rating of track 2.

2.3.4 Model-based approach

If we look at the storage complexity of user-based and item-based approaches it is O(nm), where n is size of users and m is size of tracks. This makes it unsuitable for real world scenarios where n and m are generally very high. It also has high time complexity because parsing through such matrix has high computation cost. Both the above models can be termed as memory-based models. For this reason, model-based recommender systems were introduced that make use of matrix decomposition techniques to compute recommendations more effectively.

In model-based approach, a joint latent factor space in formed by using 3 different matrices and mapping items and users. This method of forming joint latent factor space is termed as singular value decomposition (SVD). The equation of forming reduced matrix is:

$$R = U\Sigma \mathrm{V}^{\mathrm{T}} \tag{2.1}$$

Where R is the resultant reduced matrix, U is the normalized matrix eigenvectors of RRT and V is eigenvectors of RTR. Matrix Σ is diagonal matrix with non-negative real numbers called singular values which are square roots of eigenvalues. This reduced matrix is then used for all operations to form recommendation which now has lessened size of the matrix.

2.4 Machine Learning

In the realm of artificial intelligence (AI), machine learning is a well-studied issue. It focuses on problem-solving by allowing computers to learn from data rather than being expressly programmed for a specific job. Because many real-world chores are similar to this, this is a fundamental strength of this notion. Too difficult for humans to tackle effectively, if at all. The purpose of machine learning is to create algorithms that can detect patterns in data, supplied facts, collect knowledge, and use this method when confronted with fresh situations and upcoming difficulties. There are three major

categories of machine learning algorithms as explained below

- **Supervised Learning**: Every row/sample has been assigned a label which are the outputs desired. In supervised learning process, the machine learning algorithm knows which column to make the prediction for. These labels can be categorical or continuous. As an example, predicting if an e-mail is spam or not is a supervised learning problem.

- **Unsupervised Leaning**: In this process there are no labels attached with each sample. This kind of approach is used in cases where we only need grouping of samples based on their feature values without any known labels. Clustering is a popular approach followed for unsupervised learning. For example, clustering used based on type of applications they have access to in a company is an unsupervised learning problem.

- **Reinforcement Leaning**: Reinforcement leaning is a greedy approach to solving a complex problem. These types of models are designed to predict the best possible solution at every step. They are different from supervised learning as there is no defined set of results that can be obtained. Reinforcement learning is able to solve difficult task in a better way than any other models.

For the task of automatic playlist completion in music recommendation system we use supervised learning with a label for each track that suggests if that track is present in the playlist or not which is known as task of classification. Hence, supervised learning will be further discussed in this study.

2.4.1 Basic Classifiers

This section talks briefly about the basic classifier machine learning models and their differentiation.

- **Logistic Regression**: Logistic Regression is a regression model that assumes the data to fit into a sigmoid curve. With this assumption, the coefficients are trained based on the loss function which in this case is log loss. Logistic regression is the simplest of all classifiers unlike linear regression it does not fits a linear equation into the data. Here P(y) is the probability of an outcome y = 1, also known as sigmoid curve.

$$P(y) = 1/(1 + e^{x})$$

(2.2)

- **Decision Trees**: Decision trees are the roots of all tree-based models. Decision tree is a simple explanation of how a human decision-making works. Every node in the tree splits into further 2 nodes based on values on any one particular feature. These child nodes further split into 2 different nodes each until a leaf node is reached. The decision to choose most optimal feature to split on is based on reduction obtained in Gini index also known as importance gain. A leaf node is defined as node that cannot be further split either because of less samples present than the threshold value set or maximum depth of the tree is reached. If maximum depth and fraction of original dataset in the sample is not set, decision trees tend to overfit very easily. The figure below explains formation of a decision tree to forecast possibility that a tennis game will be played or not.

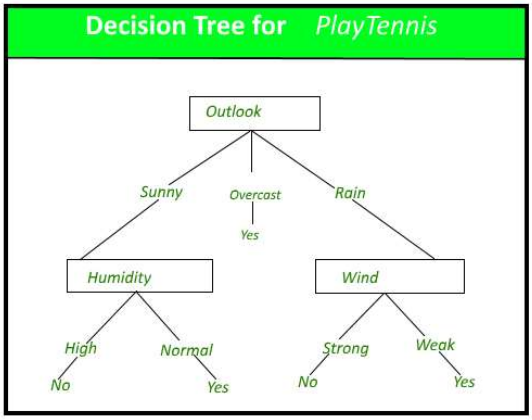


Figure 2.1: Construction of a Decision Tree

2.4.2 Ensemble Models

By integrating different models, ensemble learning improves machine learning outcomes. When compared to a single model, this method offers for improved predictive performance. The basic concept is to train a group of classifiers (experts) and then let them vote. Ensemble Learning is divided into two types: bagging and boosting. Because they incorporate numerous estimates from various models, these two reduce the variation

of a single estimate. As a result, a model with greater stability may emerge. Let's take a quick look at these two methods.

- **Bagging:** Bagging is a machine learning ensemble meta-algorithm that improves the stability and accuracy of machine learning algorithms used in statistical classification and regression. It reduces variation and aids in the avoidance of overfitting. It's most commonly used in decision tree approaches. Bagging is a subset of the model averaging strategy. The random forest model utilizes bagging ensemble where decision trees are trained on fractions of original dataset. Several decision trees make a random forest.

- **Boosting:** Boosting is an ensemble modelling strategy that aims to create a strong classifier out of a large number of weak ones. It is accomplished by constructing a model from a sequence of weak models. To begin, a model is created using the training data. The second model is then created, which attempts to remedy the faults in the previous model. This approach is repeated until either the whole training data set is properly predicted or the maximum number of models has been added. Adaboost algorithm based on boosting ensemble won the prestigious global prize develop by Schapire and Freund. It is a very popular technique that combines multiple week learners into a single strong classifier. Xgboost is more efficient version of adaboost which also works on regression task.

We will be discussing more about how these ensembles are used in automatic playlist completion task in more details in the next sections.

2.5 Deep Learning

Deep learning is the branch of machine learning that is based on neurons. They mimic how a human brain works, human brain is a network of neurons and each neuron serves its purpose that is acting as an activator. For every input it decides whether the output will be 1 or 0. Neural networks are designed in the similar way. Neural networks are networks of neurons with multiple layers in which each layer has its own weights and biases that are trained with the data. Deep learning requires a great processing power due

to which they came into application a long after they were invented. Artificial neural networks are discussed in detail in the next section.

2.5.1 Artificial Neural Network

Artificial Neural Networks are formed of artificial neurons known as perceptron or units. The Artificial Neural Networks of a system are made up of these perceptrons, which are stacked in a connection of layers. There could be any number of perceptrons in a layer based on how complex we want the model to be. An Artificial Neural Network typically contains an input layer, an output layer, and hidden layers. The input layer receives data from the outside world that the neural network must interpret or learn. The data then flows to one or more hidden layers in the network, which is then converted using trained weights and biases, provided as a useful data from output layer.

A training set is used to train artificial neural networks. Let's say you want to train an ANN how to detect a cat. The network is then given hundreds of different photos of cats in order for it to learn to recognise them. After the neural network has been sufficiently trained with photographs of cats, you must test whether it can accurately detect cat images. This is accomplished by instructing the ANN to categorise the photos it receives by determining whether or not they are cat images. The ANN's output is backed up by a human-provided explanation of whether or not the image is a cat image. Back-propagation is utilised to alter whatever the ANN learns during training if it identifies wrongly. Back-propagation is accomplished by fine-tuning the weights of the connections in ANN units in response to the error rate. This approach is repeated until the artificial neural network can properly identify a cat in a picture with the fewest potential errors.

The basic unit of ANNs in an artificial neuron unit. Output is computed in following manner.

$$y = \varphi(\, n\, \Sigma\, (j = 1) w_j x_j + b) \tag{2.3}$$

here n refers to number of rows in the input, b is bias, w signifies weight for every layer in the network, and φ is the activation function used which can be optimised.

2.5.2 Convolutional Neural Network

We use Convolutional Neural Network (CNN) primarily for image classification. A digital picture is representation of a visual with pixel values for every pixel which depicts the colour every pixel holds. A single neuron in CNN is designed to process a single patch and analyse massive quantity of data. With the network of neuron every field in the picture is covered. The data used to train CNNs are image's length, width and colours in red, blue and green channels. Layers of CNN have filters which has the same dimension as of the patch. Filter are always smaller size than the complete image and are slid across the whole image. Following are the details about layers of CNN.

Let's look at the layers used in CNNs:

- Input layer: As the name suggest, it holds the raw input.
- Convolutional layer: It computes a dot product to output volume between filters and image patches.
- Activation function layer: The activation function applied to the output is covered in this layer.
- Pool layer: This layer is added into the covnets on a regular basis, and its major goal is to lower the volume size, which speeds up calculation, saves memory, and avoids overfitting. Max pooling and average pooling are two typical types of pooling layers. The final volume will be 16x16x12 if we use a max pool with 2 x 2 filters and stride 2.

2.5.3 Recurrent Neural Network

Recurrent Neural Networks (RNN) differ from other neural networks as they train data with the temporal element. The data is presented to RNNs in sequential pattern. The RNN model is expected to learn the pattern in the sequence and then predict the next element. The key idea in RNNs is to add connections from previous iteration that remain unused. These connections can be either between neurons in the same layer or different layers. The model memorizes every piece of information from previous iterations and use them afterwards when required.

Since RNN memorize information from every sequence the memory usage is massive. There are models which work on the assumption that memory from an element very far in the sequence when arranged in chronological order is not required since it vanished with the order. These models are able to compensate for high memory usage and run

RNNs with relatively lesser computational power. Long short-term memory NN (LSTM) is one such model. The usage of these deep learning models for automatic playlist completion is discussed in further sections of this study.

2.6 Automatic Playlist Completion

An ideal MRS made for automatic playlist completion should have the ability to recommend relevant addition to the existing playlist of tracks automatically. As we discussed earlier as compared to movies and books, the length of music is relatively much shorter and hence recommendations are required more frequently than any other recommender system. For this challenge Spotify introduced Million Playlist Dataset (MPD) in 2018 (Chen et al., 2018). In this section we describe various models use for the challenge of MRS and their limitations.

2.6.1 Information Retrieval

These models use data in textual format such as track name, artist name, lyrics, etc. They are also called demographic based models (Laplante, 2008). Metadata-based Information Retrieval (IR) models are purely based on track and artist information and are relatively fast and accurate.

But at the same time, these models are very limited. For example, it is difficult to know about every song's textual information. Since IR models only consider textual information, a lot of tracks will be omitted from recommendations. Secondly, these models do not take into account users' preferences at all. Suggestions based on personal choice are more in-demand that based on item similarity.

2.6.2 Audio-based Information Retrieval

Similar to the previous method, but these models also take into account the acoustic features of the song (Mak et al., 2010; Bogdanov et al., 2011). Audio based IR model recommends songs based on what user has listened to in the past. This is one of the very first content-based approach in music recommendation systems (Aucouturier and Pachet, 2002; Logan, 2004). They measure the similarity between the songs in 3 different ways.

1. K-means Clustering with Earth-Mover's Distance – General distance between

gaussian components (Aucouturier and Pachet, 2004).

2. Monte Carlo Sampling – On top of K-means gaussian components, this method does random sampling of those distances (Logan and Engineers, 2014).
3. Euclidean Distance – This method calculates mean and variance for distances (Chordia et al., 2008).

Content-based models solves the problems in textual models as well as collaborative filtering. But at the same time, these models still lack the usage of user likes, the choice of music. Also because of the acoustic features, the number of features considered is not limited. Other information related to user demographics and also be included.

2.6.3 Collaborative Filtering Models

To take user preference into account, collaborative filtering is of a great use. The detailed study about all types of collaborative filtering models is explained in one of the earlier sections. Here we will discuss about how all approaches of collaborative filtering is utilized in MRS and their limitations.

As we know, the memory-based models make use of the user-item matrix to score recommendations for the unknown pairs. The nearest neighbour approach is highly explored in the past researches (Burke, 2002). Various model-based approaches are also invented which recommend tracks based on the reduced matric from the memory-based approaches. This reduction is done via predefined models (Adomavicius and Tuzhilin, 2005). Apart from memory-based and model-based approaches, there are hybrid methods which makes use of both the methods and are observed to perform better than both (Wang et al., 2006).

There are couple of limitations with collaborative filtering Models. Firstly, they cannot be used for popularity bias and cold start problem which is discussed in detail in another section of this study. Secondly, collaborative filtering assumes that users with similar behaviour will like the same music. This assumption is not researched well to be used in real-world data.

2.6.4 Deep Learning Models

Deep learning is explored in all types of recommendation system. Typically, recommender system uses Deep Neural Networks (DNN) for textual features like artist

name, track name, album etc, and derive their latent factors to be used by other embedded models for recommendation. These other embedded models are content-based or collaborative filtering model which work on neighbour recommendation.

One of the very first works in music recommender system to use Deep learning was (Oord et al., n.d.), where CNN was used without dropout, every song represented by fifty latent factors of music features. They utilized ReLu for activation and achieved an AUC score of 0.772. Another work using deep learning was to represent every sound clip with a matrix which is then reduced by using PCA (Wang and Wang, 2014). They used deep-belief network (DBN) coupled with stochastic gradient descent. This model performed fairly well on cold-start problem with AUC score of 0.478. (Sachdeva et al., 2018) proposed another approach that uses attentive neural network that uses one-hot encoding to represent songs then create a probability distribution for the next selection of song.

Even after an increasing growth in the usage of deep learning for music recommendation systems, there are some challenges faced. Firstly, there is no fixed evaluation metrics defined that is used by all the algorithms. Some algorithms provide better accuracy and some other provide a better AUC score which makes it difficult to pick the best. Secondly, DL models are not easily interpretable. Transparency has been a major issue since the beginning, due to which there is no reasoning for predictions. Although there are some solutions proposed to overcome transparency which combines decision trees with CNN (Zhang et al., 2019). Thirdly, proposed approaches of using deep learning in music recommender system do not take into account the sequential behaviour of the dataset.

2.7 Summary

Music recommendation system is one of the very trending topics for research for the past 5 years. Evolution of music recommender system has been described in this section. MRS started with basic information retrieval models which solely depend on track information. The state-of-the-art music recommender system now uses deep learning to gather insights from textual information and combines it with CF or CBF models. We also saw that there are still some challenges that needs to be encountered while recommending next song in the playlist. The performance from existing deep learning models can be further improved if we take into account the sequential nature of the playlist.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

Automated Playlist Continuation (APC) is getting more popular as people listen to music online. While there has been substantial progress in recent years, the majority of the methodologies presented are tested on exclusive datasets, making open collaboration and convention impossible. By undertaking standardised assessments of playlist continuance trends, the 2018 ACM RecSys Challenge intends to close this gap. At the centre of this ambition is the Million Playlist Dataset (MPD), which was launched with the support of Spotify. MPD is the largest publicly accessible dataset of its sort, with over 2.2 million songs and 1 million playlists. The study's goal is to create a playlist continuation version that can suggest suitable following songs for each playlist. For the evaluation, a different set of 10K test playlists is used, with a few songs withheld. The duration of the test playlists ranges from zero (cold start) to a hundred tracks. This mimics a real-world scenario in which the APC model must perform correctly at all degrees of playlist completion. This study's methodology is mostly based on a two-level structure. The first stage seeks to build a big selection of tracks with high recall value for each playlist, and the second stage rates only the tracks generated from first set and select top k. Below are the steps involved for this research.

3.2 Methodology

Step1: Data Cleaning & Understanding

Data understanding and cleaning is important part of a machine learning research. In most research works, data is collected from publicly available sources, which in this case is the MPD published by Spotify and available publicly as an open contest at AiCrowd. Data cleaning steps will be carried out on MPD like removing data sparsity, filling or removing missing values, feature reduction, correlation analysis between features etc.

Step 2: Exploratory Data Analysis

Exploratory Data Analysis (EDA) is carried out to gain hidden insights from the dataset. Univariate, bivariate and multivariate analysis will help in understanding relation between multiple features w.r.t target feature. For visual examination, various charts will

be used such as bar chart, heat map, scatter plot, etc. These charts will help to understand trends between different features of the dataset.

Step 3: Pre-processing and Transformation

Pre-processing is required to make data ready for modelling. In the pre-processing steps categorical features will be transformed using one-hot encoding. Features data-type will be restored to the one supported by machine learning model. In addition, outliers will be found and removed from every column in the dataset.

Step 4: Models:

Deep learning, Clustering and Collaborative Filtering are some models to be trained for this research. Every model will be tuned using grid search cross validation. The approach followed will be to first build a baseline model and keep iterating the parameters until we achieve the best results. Different models like xgboost, random forest etc. will also be explored to make sure we use best fit model.

Step 5: Evaluation:

Evaluation is done on test dataset containing 10k playlists of varying lengths. Three different metrics for evaluation are used to test different features of the solution: Recommended songs clicks, Normalized discounted cumulative gain (NDCG) and R-precision. More detail about each of the evaluation metric and what they test is present in "Evaluation Metrics" below.

Step 6: Results & Conclusion

Consolidated results from deep learning, clustering, xgboost and collaborative filtering models will be generated. These results will give insights about which set of models performs the best and will be compared with actual test set results available to evaluate at AiCrowd (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021).

3.3 Dataset description

This research will consume the Million Playlist Dataset (MPD) published by Spotify and available publicly as an open contest at AiCrowd. This data is appropriate for the research because it contains playlist of varying lengths and cold-start problem as well.

Spotify being the primary choice for online music streaming for many users all over the world, it has ample number of tracks for all popular languages. This collection of 1 million playlists, drawn from Spotify's over 4 billion public playlists, has over 2 million distinct tracks by approximately 300,000 artists, making it the world's largest publicly available dataset of tracks and playlists. Between January 2010 and November 2017, public playlists generated by Spotify users in the United States were included in the dataset. Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every track. Playlists are filtered to remove tracks with offensive content. Below table has description for features of playlist and track.

Table 3.1: Playlist attributes

Attribute	Description	Type
pid	The MPD ID of this playlist is its playlist id. This is a number that ranges from 0 to 999,999.	Integer
name	This is the title of the playlist.	String
description	This is the playlist's description. Note: Because for playlist created by users, description is Spotify's new feature, the majority having missing description.	String
modified_at	When was the last time the playlist was updated? Times are rounded to nearest hour on date the playlist was last updated, which is midnight GMT.	Timestamp
num_artists	The number of distinct artists represented in the playlist's songs.	Integer
num_albums	The total number of albums that each track in the playlist has.	Integer
num_tracks	The playlist's total amount of tracks.	Integer
num_followers	The number of people who followed this playlist when the MPD was generated.	Integer
num_edits	The total number of times playlist was edited. Tracks updated into playlist within a two-hour time frame are assumed to be part of a single edit.	Integer

duration_ms	In milliseconds, the duration of all the tracks added in playlist.	Numeric
collaborative	Whether the playlist is collaborative or not. A collaborative playlist can have tunes contributed by multiple people.	Boolean

Table 3.2: Track attributes

Attribute	Description	Type
pid	The MPD ID of playlist the track belongs to. This is a number that ranges from 0 to 999,999.	Integer
track_name	The title of the song.	String
track_uri	The track's URI on Spotify.	String
album_name	The album's name is the title of the track.	String
album_uri	The album's URI on Spotify.	String
artist_name	The major artist on the track's name	String
artist_uri	The primary artist's Spotify URI for the track.	String
duration_ms	The track's duration in milliseconds	Numeric
pos	The track's position on the playlist.	Integer

This challenge's test set consists of ten separate playlist groups, each having 1,000 playlists. Each group represents a distinct task in terms of completing a playlist:

i. Recommend songs for a playlist based solely on its title
ii. Recommend songs in a playlist based on the title and first track.
iii. Given a playlist's title and the starting five songs, predict the playlist's tracks.
iv. Recommend songs for a playlist based on the first five songs (without title)
v. Recommend songs for a playlist based on the title and the first 10 songs.
vi. Recommend songs for a playlist based on the first ten songs (without title)
vii. Recommend the first 25 songs for a playlist based on the title.
viii. Recommend songs for a playlist based on its length.
ix. Recommend the first 100 songs for a playlist based on the title.
x. Given a playlist's title and 100 random music, predict the tracks for it.

The purpose of this division is to imitate the playlist continuation model's production environment. These playlists will put the model's ability to handle all stages of playlist construction to the test. We will also create a validation set similar to the test set by sampling playlist from train set.

3.4 Data pre-processing and Transformation

In this step we pre-process the data and make it ready for modelling. First step to pre-process MPD is to remove tracks and artist that appear very rarely, because analysing rare samples does not provide statistically significant patterns. For reducing this data sparsity, we filter out tracks that appear less than 6 times and artists that appear less than 4 times. After pruning, the number of tracks is 26% of original count which is 598,293 and number of artists is 155,942 (53%). We would also validate our model on full data to test the hypothesis that if these rare samples provide any significant improvement in accuracy. For playlist titles, we set maximum length to 25 and include only alpha-numeric as well as few special characters (/<>+-).

3.5 Models

We will focus on modelling deep neural networks that include an input layer, a collection of hidden layers, and an output layer in this research. Deep learning model will be used with scores/utility from neighbour-based collaborative filtering models. For activation ReLu function may be used because of its simplicity in optimization, in addition exploring Sigmoid and tanh functions as well. For solving cold-start problem, we plan to explore 2 approaches: dimensionality reduction with SVD for playlist titles only. Adam optimizer can be used for the deep learning model and RNN model with learning rate of 0.005, which will be tuned using cross validation. For Collaborative filtering, 4 neighbour-based (track-track, word-track, album-track, artist-track) or WRMF model can be used. These set of models are expected to produce best results as they are state-of-the-art and are proven to perform better than older ones. Algorithms will be redefined and updated to best suit the use case and will be compared with original to see if they bring any significant changes in accuracy.

3.6 Evaluation Metrics

The R-precision, defined as:

$$R - precision = \frac{|T \cap R|}{|T|}$$

where T is the set of ground truth (holdout) tracks, and R is set of recommended songs.

The Normalized discounted cumulative gain (NDCG), defined as:

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

where:

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

$$IDCG = 1 + \sum_{i=2}^{|G|} \frac{1}{log_2(i+1)}$$

The Recommended Songs CLICKS metric, that mimics a Spotify feature for track recommendation where ten tracks are presented at a certain time to the user as the suggestion to complete the playlist. These metric captures required number of clicks or refreshes before a relevant track is found, and is defined as:

$$CLICKS = \frac{argmin_i\{R_i \,:\, R_i \in G\} - 1}{10}$$

While NDCG and CLICKS were calculated based on the track-level agreement between the holdout tracks and the recommended tracks, R-precision was calculated on the artist level agreement. In other words, it was considered sufficient if the artist of a recommended track matched the artist of a holdout track.

3.7 Required Resources

Required hardware and software resources for this research are:

Hardware or Environment Requirements:

- Data Storage – ASW S3 or Google Drive
- Version control system – Bitbucket or Github
- A Laptop
- Processor – Intel Core i3 or higher
- RAM – 6 GB or more
- GPU – Nvidia Graphics Card or equivalent
- OS – 64-bit Windows 10/macOS Catalina or equivalent
- Kernel - AWS Sagemaker or Google Colab or equivalent

Software Requirements:

- Python
- Python libraries
    - Pandas, Numpy, Seaborn, Matplotlib, Scipy, Statsmodel, Keras, Scikit-learn and Tensorflow
- Anaconda
- Jupyter
- Git

* Note: all latest available versions

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

The task of automatic playlist continuation requires a recommender system that is accurate as well as versatile. The million-playlist dataset from Spotify provides us playlist that have number of songs ranging from 0 to 100. This section of the study talks about the approach used to solve the problem. This study makes use of different types of features from the playlist and songs like title-embeddings and sequential-embeddings. The implementation begins with EDA on million-playlist dataset, followed by data cleaning, pre-processing, model building and ends with model evaluation step. Recurrent neural network is the uniqueness about our approach which take into account the sequential nature of the playlists. The evaluation will be done on cold-start as well as warm-start dataset, based on three different evaluation metrics. We conclude our findings in the final section of this chapter.

4.2 Dataset Description

The dataset used in this study is Spotify's Million playlist Dataset (MPD). The data contains 1 million playlists created during the period of Jan 2010-Oct2017. It contains playlists with 2.2 million unique tracks and 295,000 unique artists. The dataset is publicly available since 2018.

4.3 Exploratory Data Analysis

Since the dataset is massive and given lack of resources, we sampled 100k playlist from the dataset and completed out visualization steps.

4.3.1 Number of tracks in playlist vs song features

From Figure 4.1, we can see that the number of artists as well as number of albums in the playlist increases with increase in number of tracks in the playlist. This was as expected. The number of playlists with specific number of tracks in the playlist is maximum around 20 tracks.

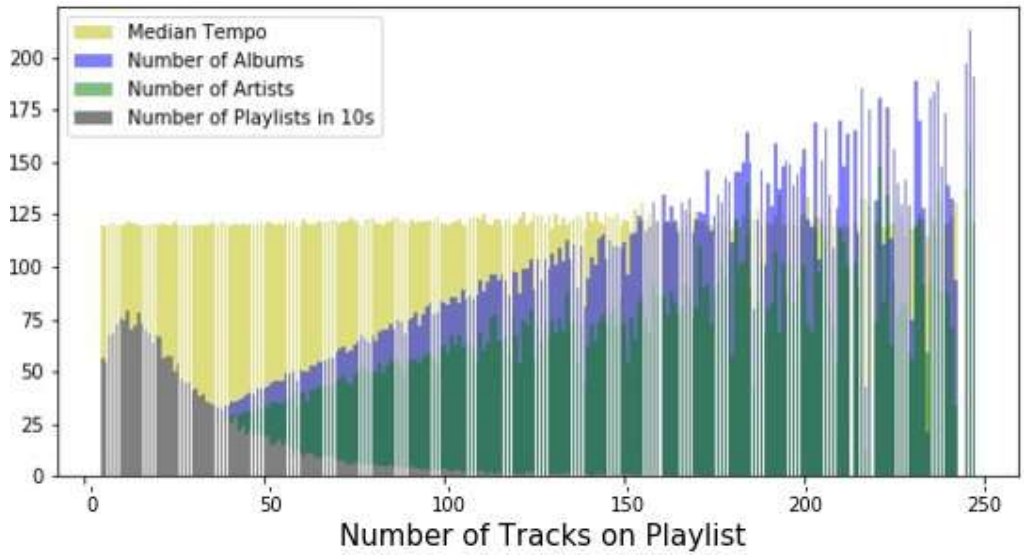


Figure 4.1 Number of tracks in playlist vs song features

In the next figure, Figure 4.2, we observe the variation of loudness in the tracks as compared to number of tracks in the playlists. Loudness follows almost a constant trend.

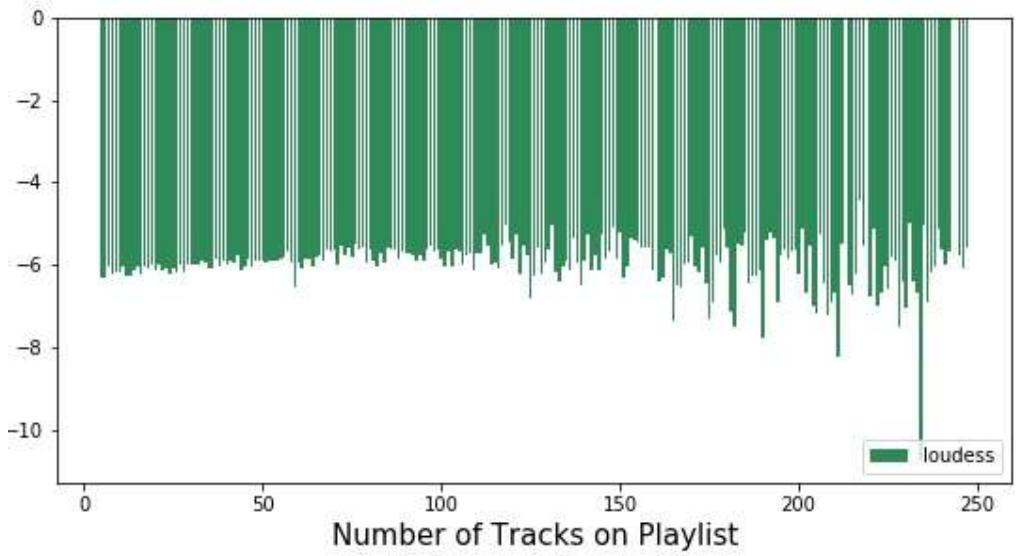


Figure 4.2 Number of tracks in playlist vs loudness

Next, we compare track's energy, danceability, speechiness, etc. with number of tracks in the playlist. These amenities have very low variance and do not deviate much on increasing number of tracks in the playlist.

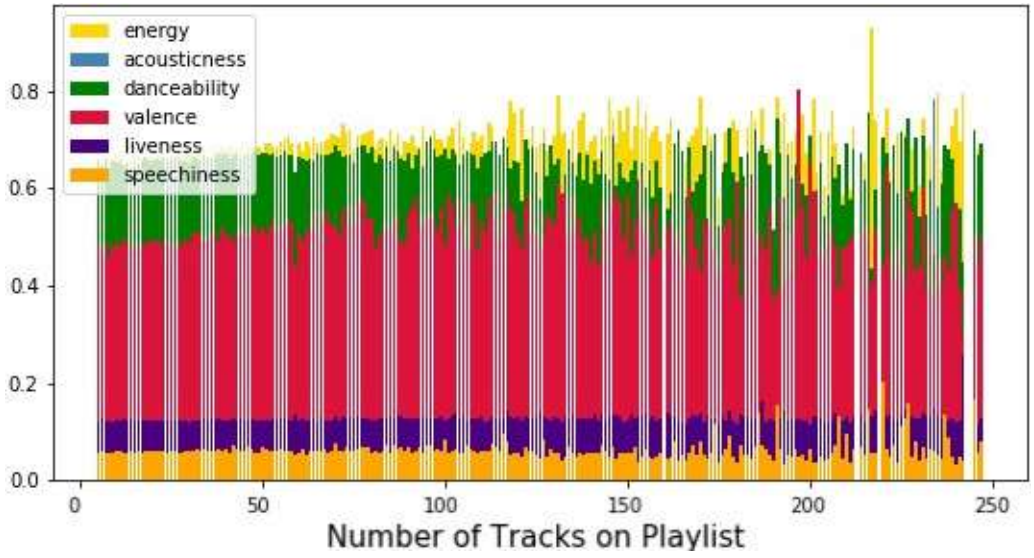


Figure 4.3 Number of tracks in Playlist vs Acoustic features

Note, the goal here is to validate if playlists with few or very large number of tracks have different acoustic features as compared to other. This validated our hypothesis was wrong and there is no such trend. We also see that on an average most of the songs are with high energy and high danceability with relatively less speech.

In the next figure, we observe the melodic and harmonic behaviour of all songs in the playlist combine. We were able to fetch this data from Spotify api (Chen et al., 2018). In figure 4.4, we compare the combined median key and mode (modus) of tracks in the playlist.

As we observe in the below figure, F-major (5) comes out to be the median key for most of the playlist and 1 is the popular mode for tracks in the playlists.

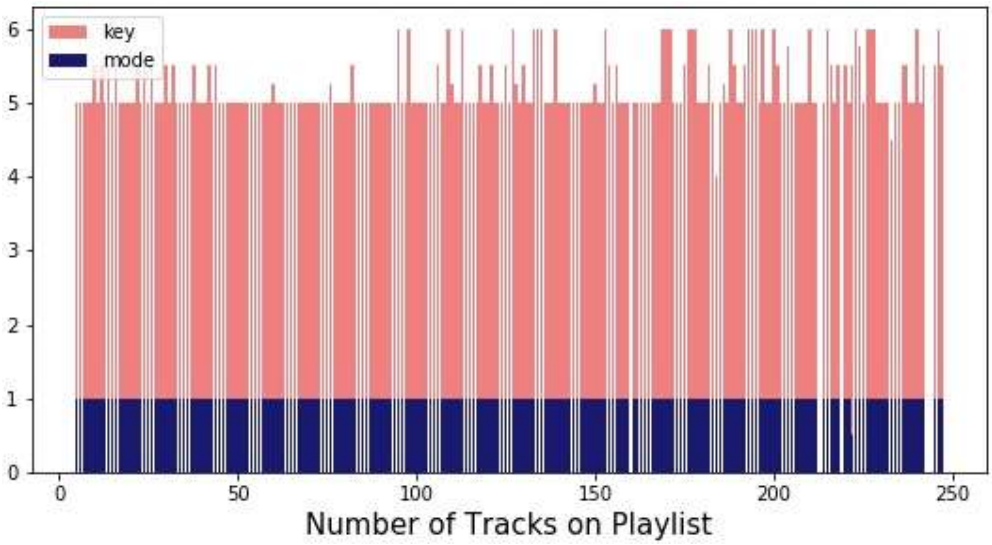


Figure 4.4 Number of Tracks in Playlist vs Harmonic features

4.3.2 1000 Playlist Dataset

We took out another 1000 playlist dataset for training of out models. As we are training a RNN, 1000 playlist are sufficient enough number to generalise pattern in sequences. More accuracy can be captures with more training data.

In figure 4.5, we observe the frequency of number of artists and number of albums in the playlist. Similar to figure 4.1, this confirms our sampled dataset also forms maxima around 20.

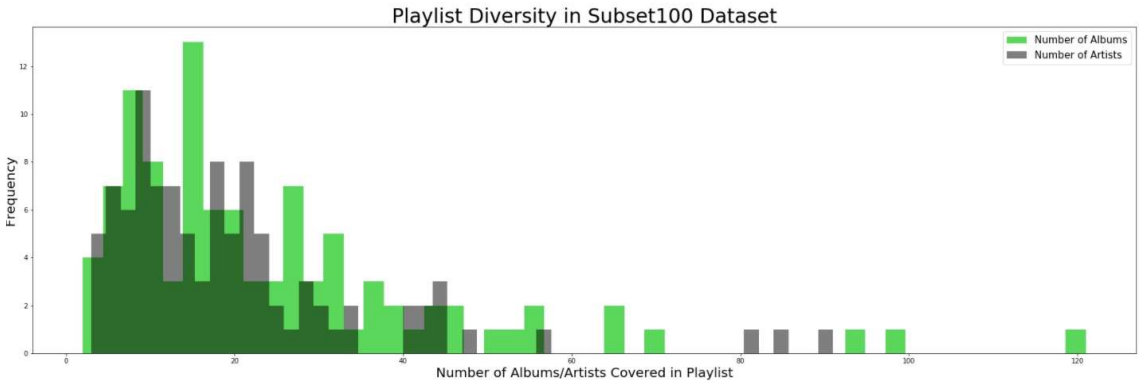


Figure 4.5 Number of Albums/Artists vs Frequency

4.3.3 Artists and tracks

Following plot tells us about the most played artists from the original dataset.

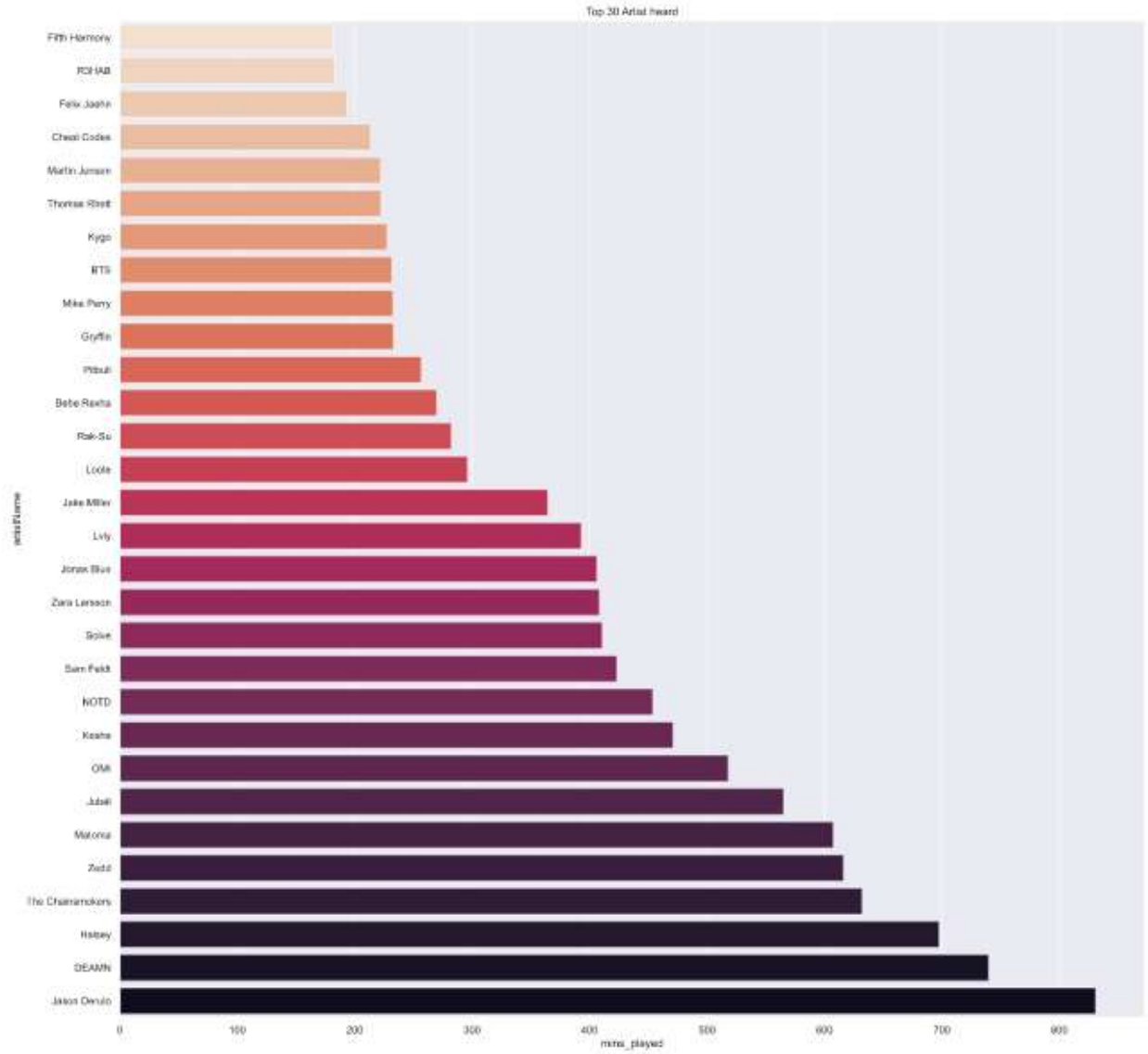


Figure 4.6: Most played artists

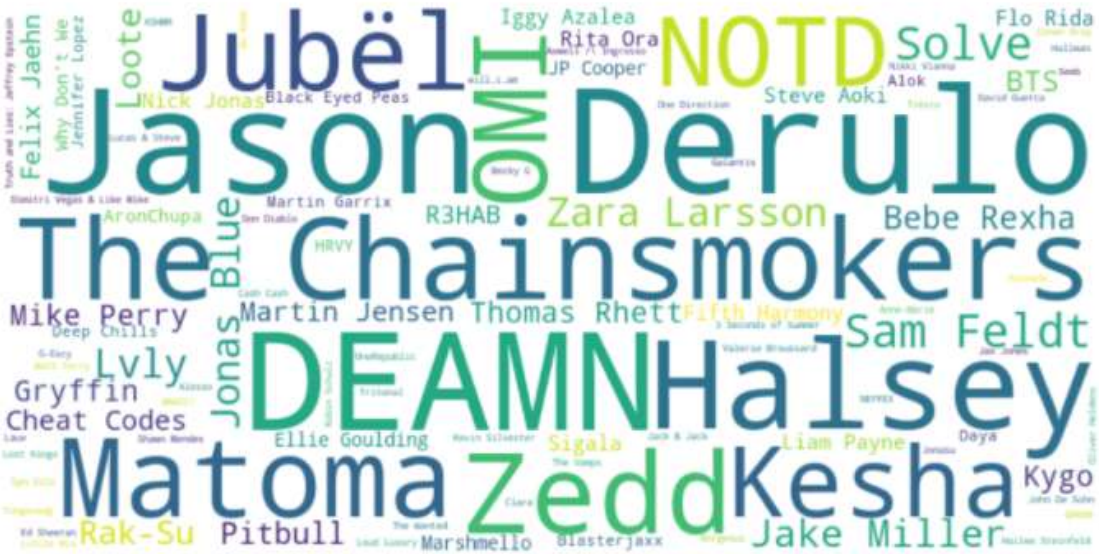


Figure 4.7: Word cloud to show top 100 artists

4.4 Data Cleaning

There are two dataframes to perform data cleaning operations on, first one is songs dataset (df1) and second one is playlist dataset (df2). We will first look at the columns of both the dataset and being with data cleaning operations.

Df1.columns

```
Index(['Unnamed: 0', 'name', 'endTime', 'artistName', 'trackName', 'msPlayed',
       'datetime', 'danceability', 'energy', 'key', 'loudness', 'mode',
       'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'type', 'id', 'uri', 'track_href', 'analysis_url',
       'duration_ms', 'time_signature'],
      dtype='object')
```

Figure 4.8 Columns in df1

Df2.columns

```
Index(['Unnamed: 0', 'id', 'spotify_id', 'list_name', 'list_id', 'song_id',
       'song_name', 'artist', 'popularity', 'release_date', 'energy',
       'liveness', 'tempo', 'speechiness', 'acousticness', 'instrumentalness',
       'danceability', 'duration_ms', 'loudness', 'valence', 'mode', 'key'],
      dtype='object')
```

Figure 4.9 Columns in df2

4.4.1 Dropping Unwanted and Duplicates

There are many unwanted columns in our dataset which might not be of any use for the analysis. We will drop 'Unnamed', 'name', 'uri', 'track_href', 'analysis_url', 'id' and 'spotify_id'. After dropping the above columns, the dataset now has 19 and 20 columns respectively.

Next, we will drop the duplicate rows present. Since there are multiple entries of a single song in songs dataset, we will keep the last and drop the rest from the dataset. Similarly, there could be multiple entries of a (list, song) pair that we can remove and keep only the latest.

After dropping the duplicates as mentioned above, we have 30% reductions in playlist dataset and 45% reduction in songs dataset. This confirms that there more than 1 entry for a song or a (list, song) pair.

4.4.2 Null Check

We then perform null check on both the dataframes to find out number of rows with null values w.r.t each column. This can be done using .isnull() function of the pandas dataframe object which gives us Boolean for every row whether it is null or not. We sum the result on axis=0 to get total nulls for each column using .sum() function. Following are the results obtained after null check.

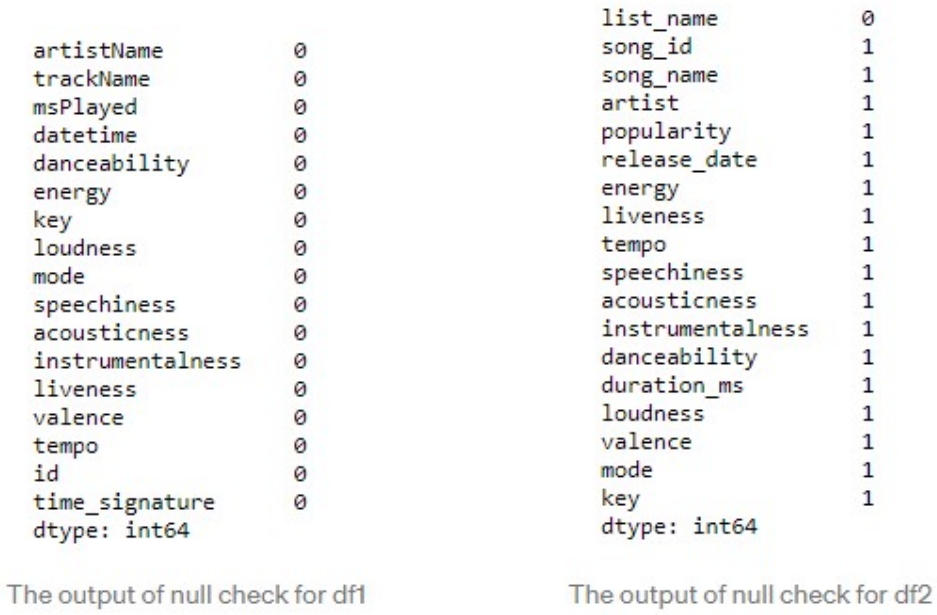


Figure 4.10: Number of null values in each column

From the results we see that there is only one null value in playlists dataset and hence that could be dropped straight away.

4.4.3 Data Types

Next step is to check the data types of all the columns and if there is a data type mismatch, we will correct it. To see data type of all columns at once we have .dtype attribute of pandas dataframe object. We get following results on checking both dataframes datatype.

This is the original dataframe taken from publicly available Spotify's database. It can be confirmed from above figure that there is no datatype mismatch present in both the dataframes.

artistName	object
trackName	object
msPlayed	int64
datetime	object
danceability	float64
energy	float64
key	float64
loudness	float64
mode	float64
speechiness	float64
acousticness	float64
instrumentalness	float64
liveness	float64
valence	float64
tempo	float64
id	object
time_signature	float64

dtype: object

the output of df1.dtypes

list_name	object
song_id	object
song_name	object
artist	object
popularity	float64
release_date	object
energy	float64
liveness	float64
tempo	float64
speechiness	float64
acousticness	float64
instrumentalness	float64
danceability	float64
duration_ms	float64
loudness	float64
valence	float64
mode	float64
key	float64

dtype: object

The output for df2.types

Figure 4.11: Data types

4.4.4 Distribution Check

To check the distribution of values in numerical columns we will look at their range (min, max) values. These can be found by using .min() and .max() functions along with axis=0. The following are the results obtained after these operation

artistName	will.i.am
trackName	Между нами любовь
msPlayed	3688090
datetime	2021-05-20 09:43:00
danceability	0.965
energy	0.996
key	11.0
loudness	-0.928
mode	1.0
speechiness	0.893
acousticness	0.995
instrumentalness	0.959
liveness	0.967
valence	0.97
tempo	213.531
id	7zv0i1f8kTy8hPapkm28kz
time_signature	5.0

dtype: object

The output of d1.max()

artistName	$teev
trackName	#thatPOWER
msPlayed	0
datetime	2020-05-12 08:49:00
danceability	0.0
energy	0.00344
key	0.0
loudness	-37.084
mode	0.0
speechiness	0.0
acousticness	0.000003
instrumentalness	0.0
liveness	0.0
valence	0.0
tempo	0.0
id	009ImBOrIUlWgla8U05RAC
time_signature	0.0

dtype: object

The output of df1.min()

Figure 4.12: min max values of df1

It is evident from the above observation that all the column has values in acceptable range and there is no value problem with our dataset.

4.5 Data Pre-processing

There are a lot of textual features in our dataset like track title, artist name, playlist name, etc. For these textual features we make use of word2vec embedding (Mikolov et al., 2013). We will train word2vec model on this textual information using the hyperparameters set by Gensim implementation (Rehurek and Sojka, 2010). After forming embedding for all textual features separately we concatenated them into one matrix. Combined with other numerical features, they are fed to our model which we will talk about in next section.

4.6 Modelling

The problem of automatic playlist completion can be thought of as predicting next word to be typed (Language modelling), similarity being the next word is decided based on the usage of previous words in the sentence or word sequence and usage of similar sentences by other users. For that reason, we train RNN in this study. The algorithm aims to get minimum cross entropy considering it as cost function. The cross-entropy loss functions the difference between observed and learned probability of target variable.

The RNN trained following below steps to compute prediction for a sequence as in figure 4.11:

- Calculate log odds (logits) for every seed in the sequence.
- Combine all log odds by taking average
- Rank tracks according to their log odds score.

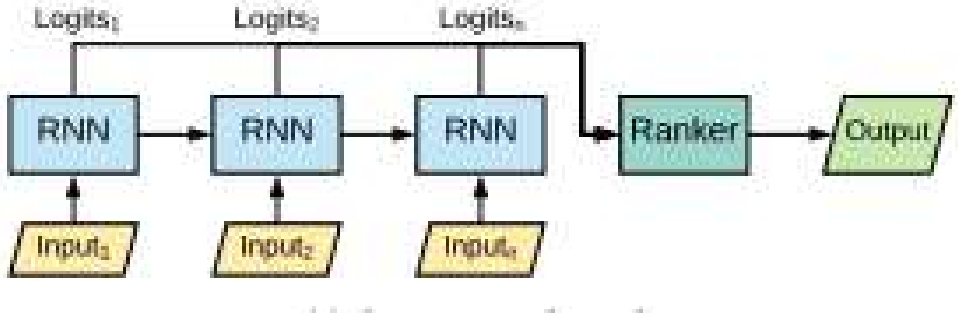


Figure 4.13: Ranking tracks based on Logits

4.6.1 Training word2vec

An additional model to be used for a cold start problem, i.e., no tracks but just the playlist title in the input is title2rec model (t2r). Here we transform the words into vectors and compute cosine similarity between new title and titles in the training dataset. Below (Figure 4.12) is the graphical representation of the model.

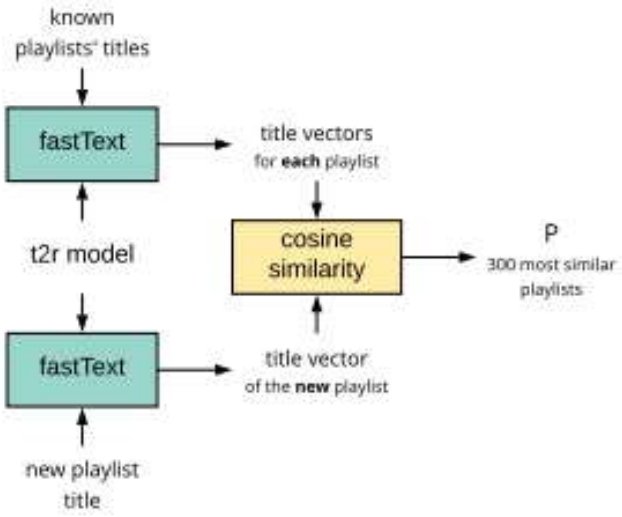


Figure 4.12: Title2rec for cold start problem

4.6.2 Tunning

We trained our RNN model described in the above approach on our sampled training dataset. For tuning the hyperparameters, we used grid search cross-validation technique. The parameters that we trained are as follows:

- Optimizer {ADAM, RMSProp and Gradient}
- Hidden layer {50, 100}
- Number of steps {10, 20}

We evaluated these trained models for all combinations of above parameters. Evaluation was done my measuring R-Precision, NDCG and click metrics as discussed in the Evaluation section of research methodology. We observed that ADAM optimizer performs better than the rest when other hyper-parameters are fixed. Increasing number of steps from 10 to 20 does not provide a significant gain in R-precision. After getting the results for all combinations, we decided to go with these parameters: learning rate =

1, optimizer = ADAM, number of steps = 10, hidden layer = 50. Further details on the results and future improvements have been explained in next sections.

Table 4.1: RNN model tuning results

Optimizer	Hidden Layer	Steps	R-precision
ADAM	100	20	0.1739
ADAM	100	10	0.1742
Gradient	100	10	0.1566
ADAM	50	10	0.1745
Gradient	50	10	0.1543

CHAPTER 5

RESULTS AND EVALUTION

5.1 Introduction

This chapter of the study talks about the output achieved from the trained RNN model, evaluation of output and compare it with some of the existing state-of-the-art automatic playlist completion techniques. Evaluation is performed on the test dataset provided in RecSys challenge from Spotify along with the training dataset. The test dataset has 10 different types of playlists with varying lengths of number of tracks present. We will talk in detail about how our model has performed on each of those types.

5.2 Results

Table 5.1 provides the set of results that were obtained on the test set from our trained RNN models.

Table 5.1: Results on test dataset

Optimizer	Epoch	R-Prec	NDCG	Click
Gradient	1	0.1417	0.1621	4.2012
Gradient	2	0.1500	0.1678	3.9435
ADAM	1	0.1572	0.1718	3.9522
ADAM	2	0.1532	0.1699	4.1342

It is evident from the above results that adam opitmizer with 1 epoch performs better than all of them for majority of the metrices. It should also be noted that for the click metric gradient optimizer seems to be performing better than ADAM. Since training these models is computationally very expensive, we could not carry out more experiments that could have contributed towards a better performing model. Training each epoch took around 16 hours even on the reduced dataset.

The model performed as per the expectations, we wanted to build a model that traps sequential nature of the playlist and it seemed to be performing good. We also observed that on increasing number of tracks as the input, the performance was increasing. We

also observed that playlist even with just a single track have significantly good performances than playlists with just the titles. On comparing with the existing state-of-the-art models, we see that our model is not performing as per the standards but our performance is comparable and the trained RNN model has potential to overcome state-of-the-art model if it is trained on the complete training dataset given that we have the time and resources to do so. Below table 5.3 has performance results of other models on same metrics.

Table 5.2: state-of-the-art results

Model	R-Precision	NDCG	Click
Two-stage model CNN + Collaborative filtering (Volkovs et al., 2018)	0.2241	0.3946	1.7839
Multimodal Collaborative Filtering CNN + encoders (Yang et al., 2018)	0.2234	0.3932	1.8952
Hybrid model with LightFM + xgboost (Rubtsov et al., 2018)	0.2153	0.3846	1.7818

Another point to note here is that our model performs poorly according to click metrics where these state-of-the-art models have 1.7-1.8 score, our model got 3.9522. This metric increases significantly as we increase number of playlists in the dataset from 1000 to 2000. This again validates that our model has potential to perform much better than these results after it is trained on complete dataset of playlists.

5.3 Summary

From the above obtained results, we learn that the RNN model performs poorer than state-of-the-art models with respect to click metrics and all other metrics. The performance can be better once we have the computational power to train it on the complete dataset. Our model was able to capture the relevant sequential information which was the goal here.

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATION

6.1 Introduction

The task of automatic playlist completion is a difficult one given the dataset is massive. In addition, there is a large variety of playlists based on users likes and behaviour. In this chapter we will talk about the conclusion that we have derived from our research and future works that can be performed to improve the recommendation engine.'

6.2 Conclusion

In this study we have presented an RNN model combined with word2vec to trap sequential nature of every playlist as well as find similar words from title name, track name and author name. This ensemble approach was optimized and validated on 10 varying types of set of playlists. The evaluation of our approach and comparison with other approaches was done with the help of 3 different matrices that measure accuracy as well as cumulative gain obtained from the recommendations. Word2vec and RNN models are complementing each other well with cold start issue.

6.3 Future Work

The computing time involved with training the RNN models turned out to be crucial set back in this study. RNNs require a great computational power to be able to train them on Millions of playlists. We addressed this problem by filtering a sample dataset from the original randomly and training our model on the smaller dataset. This was be further improved in future with increasing computational power every day around the globe.

We can also combine our RNN predictions with collaborative filtering models to understand neighbour choices and perform predictions. The current state of the model is dependent on previous tracks in the dataset but should also be tracking neighbour users and items. Another addition to our approach can be to utilize the lyrics of every song and do sentimental analysis.

By this we can group all tracks into categories based on the lyrics involved and make predictions according to the group. Audio information about the music in the track can also be utilized to categorize the tracks into genre.

All these approached combined with the current approach will improve upon the model diversity and most likely will increase its performance.

References

Adomavicius, G. and Tuzhilin, A., (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 176, pp.734–749.

Anon (2021) *AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files*. [online] Available at: https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge/dataset_files [Accessed 9 Nov. 2021].

Aucouturier, J.-J. and Pachet, F., (2004) Improving timbre similarity: How high's hte sky? *Journal of Negative Results in Speech and Audio Sciences*, [online] 11, pp.1–13. Available at: https://www.csl.sony.fr/downloads/papers/uploads/aucouturier-04b.pdf.

Aucouturier, J. and Pachet, F., (2002) Music Similarity Measures : What ' s the Use ? *Ismir*, p.7.

Bogdanov, D., Serrà, J., Wack, N., Herrera, P. and Serra, X., (2011) Unifying low-level and high-level music similarity measures. *IEEE Transactions on Multimedia*, 134, pp.687–701.

Burke, R., (2002) Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, [online] 124, pp.331–370. Available at: https://doi.org/10.1023/A:1021240730564.

Chen, C.-W., Lamere, P., Schedl, M. and Zamani, H., (2018) RecSys Challenge 2018: Automatic Music Playlist Continuation. [online] Available at: https://doi.org/10.1145/3240323.3240342 [Accessed 9 Nov. 2021].

Chen, L.S., Hsu, F.H., Chen, M.C. and Hsu, Y.C., (2008) Developing recommender systems with the consideration of product profitability for sellers. *Information Sciences*, 1784, pp.1032–1048.

Chen, S., Moore, J.L., Turnbull, D. and Joachims, T., (2012) Playlist prediction via metric embedding. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.714–722.

Chen, Y.-H. and George, E.I., (n.d.) A BAYESIAN MODEL FOR COLLABORATIVE FILTERING.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X. and Shah Google Inc, H., (2016) Wide & Deep Learning for Recommender Systems. [online] Available at: https://arxiv.org/abs/1606.07792v1 [Accessed 9 Nov. 2021].

Chordia, P., Godfrey, M. and Rae, A., (2008) Extending content-based recommendation: The case of Indian classical music. *ISMIR 2008 - 9th International Conference on Music Information Retrieval*, March, pp.571–576.

Covington, P. and Adams, J., (n.d.) Deep Neural Networks for YouTube Recommendations. [online] Available at: http://dx.doi.org/10.1145/2959100.2959190 [Accessed 9 Nov. 2021].

Hu, Y., Volinsky, C. and Koren, Y., (2008) Collaborative filtering for implicit feedback datasets. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp.263–272.

Kamehkhosh, I., Jannach, D. and Bonnin, G., (n.d.) How Automated Recommendations Affect the Playlist Creation Behavior of Users. [online] Available at:

http://www.midiaresearch.com/downloads/ [Accessed 9 Nov. 2021].

Laplante, A., (2008) Music information behaviour. [online] Spring, p.7. Available at: http://library.concordia.ca/services/users/faculty/bibliofile/spring2008/bibliofilespring2008-07.pdf.

Li, T., Liu, A. and Huang, C., (2016) A Similarity Scenario-Based Recommendation Model with Small Disturbances for Unknown Items in Social Networks. *IEEE Access*, 4, pp.9251–9272.

Logan, B., (2004) Music Recommendation from Song Sets. *Ismir*, October, pp.10–14.

Logan, B. and Engineers, E., (2014) June 2001A Content-Based Music Similarity Function A Content-Based Music Similarity Function. December 2002.

Mak, C.M., Lee, T., Senapati, S., Yeung, Y.T. and Lam, W.K., (2010) Similarity measures for chinese pop music based on low-level audio signal attributes. *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, Ismir, pp.513–518.

Mcfee, B. and Lanckriet, G., (n.d.) THE NATURAL LANGUAGE OF PLAYLISTS.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J., (2013) Distributed Representations of Words and Phrases and Their Compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13. Red Hook, NY, USA: Curran Associates Inc., pp.3111–3119.

Nicholas, I.S.C., (1999) Combining Content and Collaboration in Text Filtering. *undefined*.

Okura, S., Tagami, Y., Ono, S. and Tajima, A., (n.d.) Embedding-based News Recommendation for Millions of Users. *KDD*, [online] 17. Available at: http://dx.doi.org/10.1145/3097983.3098108 [Accessed 9 Nov. 2021].

Oord, V. Den, Dieleman, S. and Schrauwen, B., (n.d.) Deep content-based music recommendation. pp.1–9.

Paradarami, T.K., Bastian, N.D. and Wightman, J.L., (2017) A hybrid recommender system using artificial neural networks. *Expert Systems with Applications*, 83, pp.300–313.

Pazzani, M.J. and Billsus, D., (2007) Content-Based Recommendation Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, [online] 4321 LNCS, pp.325–341. Available at: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10 [Accessed 9 Nov. 2021].

Rehurek, R. and Sojka, P., (2010) Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, May, pp.45–50.

Robertson, S. and Walker, S., (2000) Threshold setting in adaptive filtering. *Journal of Documentation*, 563, pp.312–331.

Rubtsov, V., Kamenshchikov, M., Valyaev, I., Leksin, V. and Ignatov, D.I., (2018) A hybrid two-stage recommender system for automatic playlist continuation. *ACM International Conference Proceeding Series*, January 2019.

Sachdeva, N., Gupta, K. and Pudi, V., (2018) Attentive neural architecture incorporating song features for music recommendation. *RecSys 2018 - 12th ACM Conference on Recommender*

Systems, pp.417–421.

Teinemaa, I., Tax, N. and Bentes, C., (n.d.) Automatic Playlist Continuation through a Composition of Collaborative Filters.

Ungar, L.H. and Foster, D.P., (1998) Clustering Methods for Collaborative Filtering. [online] Available at: http://www.sims.berkeley.edu/resources/collab/ [Accessed 9 Nov. 2021].

Volkovs, M., Rai, H., Cheng, Z., Wu, G., Lu, Y. and Sanner, S., (2018) Two-stage model for automatic playlist continuation at scale. *ACM International Conference Proceeding Series*.

Wang, J., de Vries, A.P. and Reinders, M.J.T., (2006) On Combining User-based and Item-based Collaborative Filtering Approaches. *Proc. of the 27th Symposium on INFORMATION THEORY in the BENELUX*, pp.501–508.

Wang, X. and Wang, Y., (2014) Improving content-based and hybrid music recommendation using deep learning. *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, pp.627–636.

Xiong, R., Wang, J., Zhang, N. and Ma, Y., (2018) Deep hybrid collaborative filtering for Web service recommendation. *Expert Systems with Applications*, 110, pp.191–205.

Yang, H., Jeong, Y., Choi, M. and Lee, J., (2018) MMCF: Multimodal collaborative filtering for automatic playlist continuation. *ACM International Conference Proceeding Series*.

Zarzour, H., Jararweh, Y. and Al-Sharif, Z.A., (2020) An Effective Model-Based Trust Collaborative Filtering for Explainable Recommendations. *2020 11th International Conference on Information and Communication Systems, ICICS 2020*, pp.238–242.

Zhang, Q., Yang, Y., Ma, H. and Wu, Y.N., (2019) Interpreting cnns via decision trees. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, pp.6254–6263.

APPENDIX A: RESEARCH PROPOSAL

Abstract

Managing and looking for songs has become increasingly important as digital music formats have grown in popularity. Though successful music information retrieval (MIR) algorithms have been developed in the previous ten years, music recommender systems are still in their infancy. As a result, this study examines a broad framework as well as cutting-edge techniques to music recommendation. 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. Participants listened to music more than any of the other activities, according to previous study. Additionally, the music recommender will assist users in filtering and discovering songs based on their preferences. A smart music recommendation system should be able to detect preferences automatically and produce playlists based on them. Meanwhile, the advent of recommender systems gives the music industry a fantastic opportunity to gather users who are interested in music. More importantly, it presents us with new challenges in terms of better understanding and modelling customers' musical preferences. We explore the current state of the art in solving these problems and its limits. We go over potential future directions and visions for the field's future development. The first stage of our model is tailored for quick retrieval, while the second stage re-ranks recovered candidates to maximise accuracy at the top of the suggested list.

Table of Contents

4.0 Background

Recommender systems have become a necessary component of many e-commerce and Internet-based organisations such as Google, YouTube, Facebook, Netflix, LinkedIn, Amazon, and others have used recommender systems as part of their business strategies since the Web's inception and spectacular expansion over the last decade. firms in various domains like songs, movies, articles, shopping, etc. These systems are based on user sentiments about a product or service with the goal of promoting them to other users with similar interest according to their past behaviours and tastes. YouTube, for example, offers recommendations for a user based on their viewing patterns and searches for like users. The task of automatic playlist continuation (APC) is for music recommendation systems to find tracks that fit inside a specified playlist. A big part of the APC mission is to appropriately deduce the playlist's intended purpose. This is difficult not only because of the wide range of intended purposes (assuming they exist at all), but also because of the wide range of underlying attributes or characteristics that may be required to infer those purposes. A recommender system is an essential and appealing domain in information filtering systems that seeks to forecast active users' choice or rating of a particular item. A recommender system is a sophisticated engine that uses a variety of ways to suggest the most relevant products for a group of users based on their prior actions and preferences.

Therefore, relevant and appropriate recommendations are extremely important for these businesses to attract more users into using their platform for longer period of time. Even before the digital revolution, music services relied on a variety of entry points into the music catalogue: genres, decades, hit choices, new releases/trending, what's curators/influencers, playlists by context (moods/activities), and provided means for sharing content and playlists. In contrast to other creative content, a song is a three-minute experience, and the question of what to listen to next keeps coming up. As a result, the album's historic format, which ensures a minimum permissible duration, as well as its aesthetic objective, is preserved. Listeners now have devices, apps, and algorithms thanks to the digital revolution, which allow them to better capture those listening situation opportunities and adapt to each context: rich user interfaces on PCs,

simplified user interfaces on mobile phones and in cars, voice control in the car, and smart speakers. Listeners can experience rich navigation even when interaction with screens is limited, to the point of inviting designers to create a zero interface where no interaction is possible. Higher granularity (the ability to provide numerous types of playlists, similar artists and songs), higher frequency of selection updates, a much deeper dive into the collection, and personalisation are all advantages of digital. Various platforms like Rapcaviar, Discover Weekly provides listeners with customised playlist updated every week. Spotify on the other hand, creates multiple playlists based on users most played artist, popular songs in the area and lots of other playlist categorised by genres.

The collaborative filtering algorithm has been determined to function well based on users' listening behaviour and past ratings. When combined with the use of a content-based model, the user can get a list of songs that are comparable based on low-level acoustic variables like rhythm and pitch, as well as high-level features like genre and instrument. Last.fm2, Allmusic3, Pandora4, and Shazam5 are 4 music discovery services that have successfully used these two methodologies. Meanwhile, these websites give a one-of-a-kind platform for retrieving rich and usable data for user studies. There is a lot of possibility for future expansion thanks to new discoveries in psychology, signal processing, machine learning, and musicology. As a result, this study examines a broad music recommender framework, including user profiling, item modelling, and item-user profile matching, as well as a number of state-of-the-art methodologies. We conclude and suggest a new model based on user motivation at the end of this study.

5.0 Problem Statement

In the last decade, recommender systems have been in the focus for many studies to develop new methods and improve their accuracy. Two types of recommender systems are: collaborative filtering and content-based filtering methods. Most recommender systems are based on collaborative filtering as it is feasible and easy to implement. There are two categories of collaborative filtering: memory-based (Li et al., 2016) and model-based strategies (Paradarami et al., 2017). Model-based methods use ratings to train a model and predict ratings on unrated items by users, whereas memory-based methods use similarities between items and users to predict. Researchers (Zarzour et al., 2018) introduced a model-based music recommendation system which clusters users using k-means clustering and reduce dimensionality with SVD. One of the earliest model-based collaborative filtering methods was proposed 2 decades ago (Chen and George, 1999),

which was based on the idea that users (or judges) who tend to give similar ratings can be modeled as having identical rating probabilities.

Content-based methods take into account both the profile of user's interest and description of item. Bayesian classifiers, for example, are used to assess the likelihood of a user loving an item based on its content (Pazzani and Billsus, 2007) and another research (Chen et al., 2008) introduced contest-based recommender systems that employs profitability into CF-based systems. A text similarity method was introduced (Robertson and Walker, 2000) that sets a threshold for text similarity in item description and employs adaptive filtering. These systems suffer from many challenges such as misinterpret songs that rarely appear, completing playlist with very few tracks, neglecting sequence of tracks in a playlist. To address these challenges, there are hybrid methods proposed (Nicholas, 1999) and (Ungar and Foster, 1998) which combines collaborative features and content together. Recently, researchers (Xiong et al., 2018) proposed a deep learning-based hybrid method by combining collaborative filtering and textual content. Furthermore, explainable recommendation has also been a field of research interest (Zarzour et al., 2020) where trustworthiness score which depends on users' reactions after watching recommended videos is combined with collaborative filtering.

These collaborative filtering and content-based methods are also applied in music recommender systems. Some authors (Chen et al., 2012) and (Mcfee and Lanckriet, 2011) have developed ways for modelling the transitions between songs in playlists using Markov chains. Although, recently researchers have found that sequence of songs hardly matters to listeners while the song-to-song transitions (Kamehkhosh et al., 2018) and ensemble of songs in playlist (Chen et al., 2018) seems to matter. An online competition (ACM RecSys Challenge 2018) was held for the task of Automatic playlist continuation (APC), which involves recommending songs that are likely to be added to an existing playlist. For example, the recommended songs for a list of new year songs should be other new year songs. As part of the challenge and to promote this type of research at scale, Spotify publicly released the Million Playlist Dataset (MPD) in 2018. The data is now publicly available at (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021) . Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every

track. Playlists are filtered to remove tracks with offensive content.

The methods used by participating team in the challenge were mostly based on collaborative filtering combined with dimensionality reduction or CNN models. For instance, (Yang et al., 2018) proposed a 2-phase collaborative filtering model where in first phase autoencoders is used to encode and decode playlist features and CNN is used for training just playlist titles. Another team (Teinemaa et al., n.d.) used a three-stage model, where in first stage 4 collaborative filtering models are built on 4 context track-track, word-track, album-track and artist-track. First stage is then followed by taking weighted sum of all 4 models and then completing playlist for a given number of tracks based on same album and popular tracks. The winning team (Volkovs et al., 2018) also applied a two-stage model with additional node for cold start problem. Cold start problem occurs when there are only 2 playlist features available: name & length of the playlist. The first stage employs a WRMF collaborative filtering method which is one of most known methods for implicit/binary collaborative filtering (Hu et al., 2008), since WRMF ignores order of songs in the playlist it is clubbed with a temporal CNN and Neighbour-based CF models (song-song, playlist-playlist). The second stage contains an xgboost classifier used to re-rank prediction from 1st stage and additional playlist features like average song/artist/album, song features like artist/album, pairwise playlist song features like similarity between target song and songs in the playlist and furthermore, creative features like danceability, energy, mode etc. taken from spotify audio api.

With the recent improvement in deep learning techniques, many types of powerful news-based (Okura et al., 2017), apps-based (Cheng et al., 2016) and video-based (Covington and Adams, 2016) recommendation systems are based on deep learning. To address the issue of data sparsity and cold-start recently many deep learning techniques are proposed. Another strong technique, Collaborative Deep Learning (CDL), is given, which uses Bayesian formulation to do deep representation learning for the ratings matrix. CDL allows for two-way communication between content information and ratings matrix feedback.

However, there is no study that demonstrates how to embed people and products

together. with taking order of songs into account. Also, there are no studies that combine recent deep learning models with cold start problem to build a versatile collaborative filtering model. For example, by clubbing a neighbour-based collaborative filtering model with deep learning and employing a CNN trained on playlist titles to solve for cold-start problem can produce a more effective result. There are also no studies that take number of times a song appears in playlist into consideration.

6.0 Research Questions

The aim of this research is to develop a recommender system that performs Automatic Playlist Continuation (APC). Since playlist continuation is required to be done in production environment, the recommender system should be versatile enough to work on various limitations such as a smaller number of tracks in the playlist, only playlist title etc. and hence will be tested on such a test set.

Based on the literature review, below research questions have been formulated:

- What are the most and least important playlist and track features during APC?
- What are the best performing models to recommend songs for playlist?
- What solves cold-start problem for playlist with very less features and tracks?
- What can be laid as a foundation for future work in music recommendation systems?

7.0 Aim and Objectives

The research aims to understand factors influential in music recommendation systems with the task of Automatic Playlist Continuation. The research targets development of an efficient music recommender system to be used by online music streaming websites and eventually help listeners find more tracks they may like.

Below are the research objectives:

- To perform literature review of researches in recommendation systems.
- To perform exploratory data analysis on track and playlist features, find correlation between them and perform feature generation.
- To employ deep learning, clustering and collaborative filtering models for Automatic playlist completion.
- To compare results and identify most efficient models in creating a music recommender system.
- To consolidate results from finalized models.

8.0 Significance of Study

The study is directed to develop a music recommender system that can be used in production by music streaming platforms to give useful and relevant recommendations to its users. This study will focus on developing a solution that combines recent deep learning models with cold start problem to build a versatile collaborative filtering model which as not been explored yet. The research would also be able to answer which features contribute most and least to music recommendation, this can also greatly help streaming platforms to better understand profile of the user. This study introduces a new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques with the goal of increasing the program's recommendation performance. Music recommendation is relatively a new area of research and is gaining researchers interest over the years since the business of online music streaming platforms entirely depend on their ability to keep user attracted to the platform for as long as possible and at the heart of it sits the recommendation system.

9.0 Scope of the study

Recommendation is all about broadening a listener's musical horizons beyond what they already know and enjoy. It gives listeners more navigation speed once they've exhausted all of their song/artist search capabilities. Scope of this study is to make use of the existing deep learning, collaborative filtering and clustering algorithms and to develop new such algorithms that best suits the recommender system. The evaluation of model would be carried out on the separate test set which has playlists of varying lengths, the evaluation metrics used can tell us about recall and top k accuracy of the model. The suggested models will be trained and tested to get consolidated table and arrive at a conclusion on which models performs the best and if there is a scope of further development in the models.

7. Research Methodology

7.1 Introduction

Automated Playlist Continuation (APC) is getting more popular as people listen to music online. While there has been substantial progress in recent years, the majority of the methodologies presented are tested on exclusive datasets, making open collaboration and convention impossible. By undertaking standardised assessments of playlist continuance

trends, the 2018 ACM RecSys Challenge intends to close this gap. At the centre of this ambition is the Million Playlist Dataset (MPD), which was launched with the support of Spotify. MPD is the largest publicly accessible dataset of its sort, with over 2.2 million songs and 1 million playlists. The study's goal is to create a playlist continuation version that can suggest suitable following songs for each playlist. For the evaluation, a different set of 10K test playlists is used, with a few songs withheld. The duration of the test playlists ranges from zero (cold start) to a hundred tracks. This mimics a real-world scenario in which the APC model must perform correctly at all degrees of playlist completion. This study's methodology is mostly based on a two-level structure. The first stage seeks to build a big selection of tracks with high recall value for each playlist, and the second stage rates only the tracks generated from first set and select top k. Below are the steps involved for this research:

Step1: Data Cleaning & Understanding

Data understanding and cleaning is important part of a machine learning research. In most research works, data is collected from publicly available sources, which in this case is the MPD published by Spotify and available publicly as an open contest at AiCrowd. Data cleaning steps will be carried out on MPD like removing data sparsity, filling or removing missing values, feature reduction, correlation analysis between features etc.

Step 2: Exploratory Data Analysis

Exploratory Data Analysis (EDA) is carried out to gain hidden insights from the dataset. Univariate, bivariate and multivariate analysis will help in understanding relation between multiple features w.r.t target feature. For visual examination, various charts will be used such as bar chart, heat map, scatter plot, etc. These charts will help to understand trends between different features of the dataset.

Step 3: Pre-processing and Transformation

Pre-processing is required to make data ready for modelling. In the pre-processing steps categorical features will be transformed using one-hot encoding. Features data-type will be restored to the one supported by machine learning model. In addition, outliers will be found and removed from every column in the dataset.

Step 4: Models:

Deep learning, Clustering and Collaborative Filtering are some models to be trained for

this research. Every model will be tuned using grid search cross validation. The approach followed will be to first build a baseline model and keep iterating the parameters until we achieve the best results. Different models like xgboost, random forest etc. will also be explored to make sure we use best fit model.

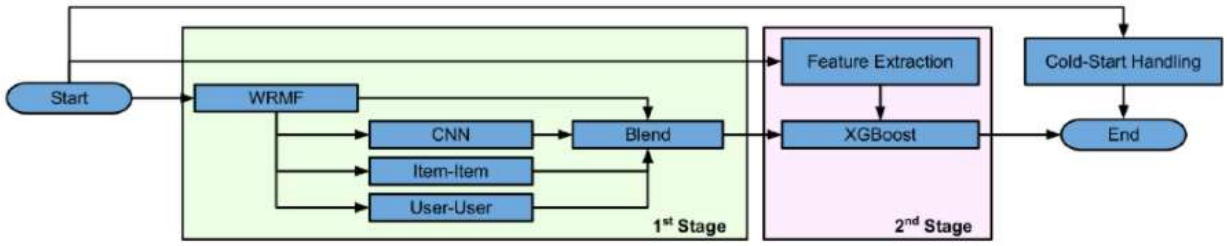
Step 5: Evaluation:

Evaluation is done on test dataset containing 10k playlists of varying lengths. Three different metrics for evaluation are used to test different features of the solution: Recommended songs clicks, Normalized discounted cumulative gain (NDCG) and R-precision. More detail about each of the evaluation metric and what they test is present in "Evaluation Metrics" below.

Step 6: Results & Conclusion

Consolidated results from deep learning, clustering, xgboost and collaborative filtering models will be generated. These results will give insights about which set of models performs the best and will be compared with actual test set results available to evaluate at AiCrowd (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021).

Figure 1: Process Flow Diagram



7.2 Dataset description

This research will consume the Million Playlist Dataset (MPD) published by Spotify and available publicly as an open contest at AiCrowd. This data is appropriate for the research because it contains playlist of varying lengths and cold-start problem as well. Spotify being the primary choice for online music streaming for many users all over the world, it has ample number of tracks for all popular languages. This collection of 1 million playlists, drawn from Spotify's over 4 billion public playlists, has over 2 million distinct tracks by approximately 300,000 artists, making it the world's largest publicly available dataset of tracks and playlists. Between January 2010 and November 2017,

public playlists generated by Spotify users in the United States were included in the dataset. Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every track. Playlists are filtered to remove tracks with offensive content. Below table has description for features of playlist and track.

Table 1: Playlist attributes

Attribute	Description	Type
pid	The MPD ID of this playlist is its playlist id. This is a number that ranges from 0 to 999,999.	Integer
name	This is the title of the playlist.	String
description	This is the playlist's description. Note: Because for playlist created by users, description is Spotify's new feature, the majority having missing description.	String
modified_at	When was the last time the playlist was updated? Times are rounded to nearest hour on date the playlist was last updated, which is midnight GMT.	Timestamp
num_artists	The number of distinct artists represented in the playlist's songs.	Integer
num_albums	The total number of albums that each track in the playlist has.	Integer
num_tracks	The playlist's total amount of tracks.	Integer
num_followers	The number of people who followed this playlist when the MPD was generated.	Integer
num_edits	The total number of times playlist was edited. Tracks updated into playlist within a two-hour time frame are assumed to be part of a single edit.	Integer
duration_ms	In milliseconds, the duration of all the tracks added in playlist.	Numeric
collaborative	Whether the playlist is collaborative or not. A collaborative playlist can have tunes contributed by multiple people.	Boolean

Table 2: Track attributes

Attribute	Description	Type
pid	The MPD ID of playlist the track belongs to. This is a number that ranges from 0 to 999,999.	Integer
track_name	The title of the song.	String
track_uri	The track's URI on Spotify.	String
album_name	The album's name is the title of the track.	String
album_uri	The album's URI on Spotify.	String
artist_name	The major artist on the track's name	String
artist_uri	The primary artist's Spotify URI for the track.	String
duration_ms	The track's duration in milliseconds	Numeric
pos	The track's position on the playlist.	Integer

This challenge's test set consists of ten separate playlist groups, each having 1,000 playlists. Each group represents a distinct task in terms of completing a playlist:

xi. Recommend songs for a playlist based solely on its title
xii. Recommend songs in a playlist based on the title and first track.
xiii. Given a playlist's title and the starting five songs, predict the playlist's tracks.
xiv. Recommend songs for a playlist based on the first five songs (without title)
xv. Recommend songs for a playlist based on the title and the first 10 songs.
xvi. Recommend songs for a playlist based on the first ten songs (without title)
xvii. Recommend the first 25 songs for a playlist based on the title.
xviii. Recommend songs for a playlist based on its length.
xix. Recommend the first 100 songs for a playlist based on the title.
xx. Given a playlist's title and 100 random music, predict the tracks for it.

The purpose of this division is to imitate the playlist continuation model's production environment. These playlists will put the model's ability to handle all stages of playlist construction to the test. We will also create a validation set similar to the test set by sampling playlist from train set.

7.3. Data pre-processing and Transformation

In this step we pre-process the data and make it ready for modelling. First step to pre-process MPD is to remove tracks and artist that appear very rarely, because analysing rare samples does not provide statistically significant patterns. For reducing this data sparsity, we filter out tracks that appear less than 6 times and artists that appear less than 4 times. After pruning, the number of tracks is 26% of original count which is 598,293 and number of artists is 155,942 (53%). We would also validate our model on full data to test the hypothesis that if these rare samples provide any significant improvement in accuracy. For playlist titles, we set maximum length to 25 and include only alpha-numeric as well as few special characters (/<>+-).

7.4. Models

We will focus on modelling deep neural networks that include an input layer, a collection of hidden layers, and an output layer in this research. Deep learning model will be used with scores/utility from neighbour-based collaborative filtering models. For activation ReLu function may be used because of its simplicity in optimization, in addition exploring Sigmoid and tanh functions as well. For solving cold-start problem, we plan to explore 2 approaches: dimensionality reduction with SVD or CNN for playlist titles only. Adam optimizer can be used for the deep learning model and CNN model with learning rate of 0.005, which will be tuned using cross validation. The size of filters for CNN model can be 3,5,6 and 9. For Collaborative filtering, 4 neighbour-based (track-track, word-track, album-track, artist-track) or WRMF model can be used. These set of models are expected to produce best results as they are state-of-the-art and are proven to perform better than older ones. Algorithms will be redefined and updated to best suit the use case and will be compared with original to see if they bring any significant changes in accuracy.

7.5. Evaluation metrics

The R-precision, defined as:

$$R - precision = \frac{|T \cap R|}{|T|}$$

where T is the set of ground truth (holdout) tracks, and R is set of recommended songs. The notation |.| denotes the number of elements in the set.

The Normalized discounted cumulative gain (NDCG), defined as:

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

where:

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

$$IDCG = 1 + \sum_{i=2}^{|G|} \frac{1}{log_2(i+1)}$$

The Recommended Songs CLICKS metric, that mimics a Spotify feature for track recommendation where ten tracks are presented at a certain time to the user as the suggestion to complete the playlist. These metric captures required number of clicks or refreshes before a relevant track is found, and is defined as:

$$CLICKS = \frac{argmin_i\{R_i \,:\, R_i \in G\} - 1}{10}$$

While NDCG and CLICKS were calculated based on the track-level agreement between the holdout tracks and the recommended tracks, R-precision was calculated on the artist level agreement. In other words, it was considered sufficient if the artist of a recommended track matched the artist of a holdout track.

8. Required Resources

Required hardware and software resources for this research are:

Hardware or Environment Requirements:

- Data Storage – ASW S3 or Google Drive
- Version control system – Bitbucket or Github
- A Laptop
- Processor – Intel Core i3 or higher
- RAM – 6 GB or more
- GPU – Nvidia Graphics Card or equivalent
- OS – 64-bit Windows 10/macOS Catalina or equivalent
- Kernel - AWS Sagemaker or Google Colab or equivalent

Software Requirements:

- Python
- Python libraries
    - Pandas, Numpy, Seaborn, Matplotlib, Scipy, Statsmodel, Keras, Scikit-learn and Tensorflow
- Anaconda
- Jupyter
- Git

* Note: all latest available versions