Clustering Neural Collaborative Filtering Recommender System for Playlist Continuation

Abstract—**This report demonstrates a music recommender system using the Million Playlist Dataset and Clustering Neural Collaborative Filtering technique. We will be implementing this system and analysing our results by comparing it to a baseline recommender system.**

Index terms—recommender system, million playlist dataset, clustering, neural collaborative filtering

I. Introduction

A. Domain

The domain of this recommender system is music. Some notable examples of platforms that demonstrate the use of recommender systems in the music domain are Spotify, Last FM and Apple Music. We will be using the Million Song Dataset provided by AI crowd [1], which gives us data on one million playlists.

B. Aim

With the increase in music diversity over the years, users have an almost infinite selection of new songs to listen to. Though great, this can make it difficult to find songs that we would enjoy which is why we would benefit from a music recommender system. On platforms such as Spotify, personalised and even non personalised recommendations are the core of the customer experience [2]. Personalised recommendations are important for these platforms as they have been known for improving user engagement and increasing company revenue, i.e. In the music industry 75% of user's experience is tightly interwoven with playlisting[3]. However, there are still issues with these systems as due to the myriad of contributing factors, accurate recommendations cannot be made simply from assuming a user has liked a song just by listening to it for 2 minutes. The goal of creating our music recommender system is to personalise recommendations to users based on their listening history and preferences. Our approach will focus on playlist continuation by suggesting songs that a user may enjoy based on the current songs in their playlist. One major issue with the Million Playlist Dataset (MPD) is scalability and lack of content features [4]. To solve these issues, we will use a Clustering Neural Collaborative Filtering method for our main recommender system and measure its effectiveness by comparing its performance to a non-personalised, popularity based method using RMSE, Hit@K ratio and NDCG as evaluation metrics.

II. Metholodology

A. Dataset

For our system we will be using data from the Million Playlist Dataset (MPD) and synthetic data we created using Spotify's API. Our MPD contains 1 million playlists from US Spotify users from January 2010 to November 2017 [1]. This includes information about each playlist including, title, id and tracks, along with their metadata. Our synthetic data uses the trackids provided by the MPD to extract audio features with a Spotify API (spotipy) [5].

B. Data Preparation

When retrieving our dataset, playlists are separated into 10,000 different json files. To start our data pre-processing we retrieve these and merge them into one big data frame for all 1 million playlists. From table 1 we can see some statistics of this dataset. Overall, from [6] it was noted that a lot of the data was already cleaned by Spotify with no missing values. We have decided for our process that we will not drop any playlists as "playlist IDs" will act as users, therefore, all rows are needed for our result. For playlist continuation, our aim is to rank song suggestions based on the current songs in the user's playlist. Therefore, we need to focus on features that will impact the relevancy of songs. Metadata about the playlist such as "num_followers", "num_edits", are examples of features that do not have any relevancy to individual song data, therefore, all columns other than "pid" and "tracks" will be dropped. Our tracks column also contains metadata about each track, we kept the "artist_name" and "track_name" as these will be outputted to our user, the position of our playlist ("pos") will be used for our train-test split and our "track_uri" is used as our track identification. For ease of matrix factorization, we replaced our unique track_uris with unique integers. As we found to have no null values in this dataset, our last step will be to remove any duplicate songs in a playlist, as in our method, the relationship between each playlist and track is only measured by interaction or not (1 or 0). Overall we have reduced the number of tracks from 66346428 to 65464776 which does not negatively affect the range or diversity of data.

When creating our synthetic data, we will be using a Spotify API and the track_uris to get audio features for each playlist. From [7] and by plotting a graph on the distribution of audio features, we found that the features with the most effect on the genre of music are:"danceability","energy","loudness","mode","acousticeness","instrumentalness", "liveness" and "valence". To find the scores of the audio features of the playlist we retrieve the audio features of the tracks in each playlist, and calculate the average of each. Reference Fig 1 for information about our cleaned dataset.

For our dataset we will be assuming that the presence of a track in the user's playlist directly correlates to them liking a song. Therefore, our dataset uses binary implicit data which comprises of feedback inferred from the user's behaviour in the form of 1 or 0. This will be taken under consideration when choosing our model.

TABLE I.

Property	*Value*	*Cols MPD*	*Cols Audio Feats*
Number of Playlists	1,000,000	track_uri; artist_name; track_name; pid; track_index; pos	Dancebility; Energy; Loudness; Mode; Acousticness; Instrumentalness; Liveness; Valence
Number of tracks	65464776		
Number of unique tracks	2262292		
Number of unique artists	287742		
Mean no of tracks per playlist	66.35		

Fig. 1. Table demonstrates final values of items after data cleaning, and our "Cols" columns represent the columns left in datasets after feature extraction.

III. Algorithms

A. Non -personalised Recommender System (Popularity Based)

In order to evaluate the effectiveness of our recommender system, we will also be implementing a baseline recommender system using a popularity based method. This will be used as a baseline when comparing and evaluating our CNCF model. This technique recommends songs to users based on how popular the song is over our 1 million playlists. This is determined by how many times each song appears in our dataset; the counts are then normalised to values between 0-1 for them to be evaluated against our personalised dataset, these are then ranked, and the highest ranked songs are outputted to the users. The big issue with this system is that it doesn't consider each users listening history or preferences so would not be as useful to our users [8].

B. Personalised Recommender System (CNCF)

For our recommender system we will be looking at Clustering neural collaborative filtering (CNCF). This is a hybrid recommender, that first uses clustering to group similar items together and then applies neural collaborative filtering to each cluster. First, we will talk why we have chosen neural collaborative filtering before discussing why we enhanced this using clustering.

Our architecture is outlined in Fig 2.

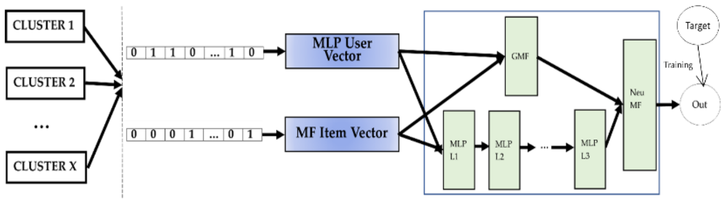


Fig. 2. Architecture of our recommender system model.

For Collaborative Filtering, matrix factorization tends to be the method used most. However, it is thought to negatively affect its performance as it uses the inner product as its interaction, therefore can output high loss and inaccurate predictions [9] on large datasets like ours. This tends to be worse for implicit feedback as the binary classification of 1 or 0 is always positive and lacks negative feedback.

Neural collaborative filtering (NCF) on the other hand can learn complex and non-linear interactions between users and items[11]. Neural networks are effective in modelling these interactions and can also work with large amounts of data [10], which is an important requirement when working with something as large as the MPD. Another reason why we would benefit from an NCF model is its ability to learn from implicit feedback [11].

However, a few issues we have with this method is 1) The cold start problem and 2) Scalability [12] hence why we decided to improve it with clustering beforehand. We decided to create a synthetic dataset of audio features, to summarise our playlist contents to help combat the cold start problem. Whilst we only use Neural collaborative filtering on the original MPD, we use clustering beforehand on our audio features dataset. By doing this, target playlists will have songs recommended from their own cluster so will have content similarities and won't only rely on results of collaborative filtering. Another advantage of using CNCF on the MPD, is that whilst neural collaborative filtering relies on matrix factorization which can become very large and sparse [10], clustering similar items into separate groups beforehand, can reduce the dimensionality and sparsity of the data as well as reduce computational costs [13]. Before implementing this, we will need to perform a train-test split on our data.

Train - test split is used for evaluating a machine learning model. For our recommender system and our playlist dataset doing a random train-test split would not be fair as the dataset may have anomalies which would not accurately reflect the user's current listening preferences if these are chosen as the points for the test set, leading to poor generalization performance [11].

However, the leave one out method is a technique used for evaluating the performance of a recommendation algorithm. For each user, we keep their latest interaction with the item as a test set, and we use the remaining as a training set for the model. The model is then used to predict the songs in the left-out playlist, and the performance of the model is evaluated based on how well it was able to predict the left-out song. The process is repeated for each user in the dataset, and the performance of the algorithm is evaluated based on the average performance across all users [11]. This method is useful for evaluating the generalization performance of a recommendation algorithm and can provide an estimate of how well the algorithm would perform on new users.

The first step is to use clustering. We will use Kmeans on our extracted audio features. We decided that 100 clusters was best for our large dataset to split it into groups small enough for our neural collaborative filtering to handle. However, with many clusters we

$$J = \sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - c_j\|^2 \tag{1}$$

need to consider that this increases the risk of overfitting [14]. Kmeans follows the formula (1) where we used Euclidean distance.

where J. k, $x_{i,}$ c_j are the criterion function, the number of clusters, the ith sample, and the centroid of jth cluster C_j

Next we have Neural Collaborative Filtering is describe in the following steps using techniques and equations from [15]:

1. Map relationships between users/playlists and items/tracks by creating a sparse one hot encoded matrix, where 1 means user u has listened to item i and 0 otherwise.

2. Playlists and Track vectors are used to create embeddings for each playlist and item. We pass our one hot encoded vector into embedding layers for our model to understand our sparse data points[10]. However, one drawback of using implicit feedback is that there is a natural scarcity for negative feedback. Therefore we created negative training examples and we use a tower architecture as this allows the model to learn more abstract features.[16]

3. Next we have General Matrix Factorisation (GMF) which uses the dot product to combine the two embeddings and uses sigmoid as the activation function as formulated in (2).

$$\hat{r}_{u,i} = a_{out}(h^T(q_i \odot p_u)) \tag{2}$$

where p_u, q_i, $\odot$, a_{out},h represents the user latent vector, the item latent vector, the dot product, the activation function, and weights of output layer respectively.

4. Multi-Layer Perceptron (MLP) uses concatenation (3) to create a feature vector for the next layers. (4) formulates the hidden layers and (5) the outer layer.

$$z_1 = \phi_1(p_u, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix} \tag{3}$$

$$\phi_l(z_l) = a_{out}(W_l^T z_l + b_l), (l = 2, 3, ..., L-1) \tag{4}$$

$$\hat{r}_{u,i} = \sigma(h^T \phi(z_{L-1})) \tag{5}$$

where z_1 is the concatenation of p_u and q_i, and W_1,b_1, a_{out},$\hat{r}_{ui}$,h,$\sigma(x) = \frac{1}{1+e^{-x}}$, represent the weight matrix, bias vector, activation function, the predicted rating, the edge weights, activation function (sigmoid) respectively.

5. We create dense layers each using ReLu activation functions. GMF and MLP are then combined before inputting them into the NeuMF layer to make our predictions [17]. By combining these it applies both their best features. [10] For our model, through hyperparameter tuning we found that our best values are batch_size = 256, 4 dense layers, learning_rate = 0.001 and training it for 10 epochs the process is shown in equations (6), (7) and (8)

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G, \tag{6}$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L), \tag{7}$$

$$\hat{r}_{u,i} = \sigma\left(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}\right) \tag{8}$$

where $\hat{r}_{ui}$,σ, h, ϕ^{GMF} and ϕ^{MLP} denote the predicted ratings, sigmoid activation function, edge weights of the output layer, last hidden layer of GMF and last hidden layer of MLP.

6. We train our model by using ADAM as our optimizer and binary cross-entropy as our log loss (9).

$$L = -\sum_{(u,i) \in \mathbb{O} \cup \mathbb{O}^-} r_{u,i} log \hat{r}_{u,i} + (1 - r_{u,i}) log(1 - \hat{r}_{u,i}) \tag{9}$$

where $\mathbb{O}$, $\mathbb{O}^-$,r_{ui} , $\hat{r}_{ui}$ are the set of observed interactions, the set of negative instances, the actual ratings and the predicted ratings respectively.

7. Before outputting our final predictions we use Kmeans and the playlist embeddings to find the nearest neighbour tracks for our target playlist. [16]

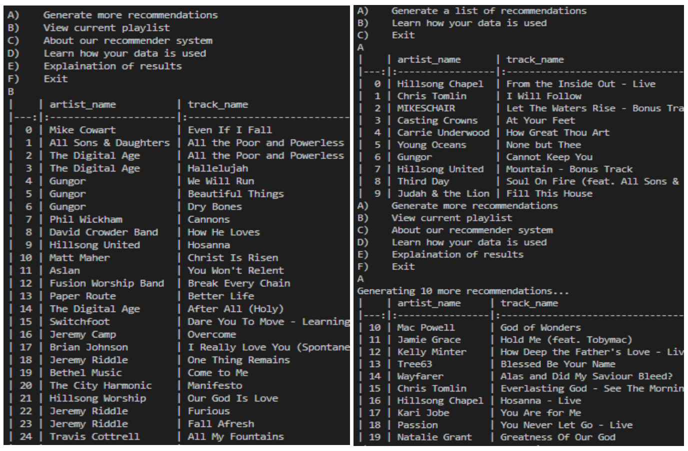
IV. Interface

A. Input Interface

In our input interface we welcome the user and ask them to choose an option from multiple choice. We enable them to either put in their user ID, Learn about User IDs, Learn about our recommender system or access it as a guest and receive non-personalised recommendations. To protect the user's privacy the input interface requires a password to access user data and recommendations. The next options allow the user to either ask for a list of recommendations or to view their current playlists. In addition, when the recommendations are given, we enable the user to either request more recommendations as well as learn more information such as how their data is used which helps with algorithmic and data transparency.

B. *Output Interface*

As our recommender system is focused on playlist continuation, we will only recommend 10 songs initially, and then the user is able to ask for 10 more songs at a time. We did this based on Spotify's app interface [18] where they only suggest a small number of songs for a playlist at a time. We display this in a table on our console interface with only the order number, the track name, and the track artist, we did not find that the prediction percentages were meaningful enough for the user as the comparisons of the top recommendations tended to be a $1x10^{-3}$ difference from each other.



```
A)    Generate more recommendations
B)    View current playlist
C)    About our recommender system
D)    Learn how your data is used
E)    Explaination of results
F)    Exit
B
|    | artist_name           | track_name
|---:|:----------------------|:-----------------------------
|  0 | Mike Cowart           | Even If I Fall
|  1 | All Sons & Daughters  | All the Poor and Powerless
|  2 | The Digital Age       | All the Poor and Powerless
|  3 | The Digital Age       | Hallelujah
|  4 | Gungor                | We Will Run
|  5 | Gungor                | Beautiful Things
|  6 | Gungor                | Dry Bones
|  7 | Phil Wickham          | Cannons
|  8 | David Crowder Band    | How He Loves
|  9 | Hillsong United       | Hosanna
| 10 | Matt Maher            | Christ Is Risen
| 11 | Aslan                 | You Won't Relent
| 12 | Fusion Worship Band   | Break Every Chain
| 13 | Paper Route           | Better Life
| 14 | The Digital Age       | After All (Holy)
| 15 | Switchfoot            | Dare You To Move - Learning
| 16 | Jeremy Camp           | Overcome
| 17 | Brian Johnson         | I Really Love You (Spontane
| 18 | Jeremy Riddle         | One Thing Remains
| 19 | Bethel Music          | Come to Me
| 20 | The City Harmonic     | Manifesto
| 21 | Hillsong Worship      | Our God Is Love
| 22 | Jeremy Riddle         | Furious
| 23 | Jeremy Riddle         | Fall Afresh
| 24 | Travis Cottrell       | All My Fountains
```

```
A)    Generate a list of recommendations
B)    Learn how your data is used
C)    Exit
A
|    | artist_name         | track_name
|---:|:--------------------|:------------------------------------
|  0 | Hillsong Chapel     | From the Inside Out - Live
|  1 | Chris Tomlin        | I Will Follow
|  2 | MIKESCHAIR          | Let The Waters Rise - Bonus Tra
|  3 | Casting Crowns      | At Your Feet
|  4 | Carrie Underwood    | How Great Thou Art
|  5 | Young Oceans        | None but Thee
|  6 | Gungor              | Cannot Keep You
|  7 | Hillsong United     | Mountain - Bonus Track
|  8 | Third Day           | Soul On Fire (feat. All Sons &
|  9 | Judah & the Lion    | Fill This House
A)    Generate more recommendations
B)    View current playlist
C)    About our recommender system
D)    Learn how your data is used
E)    Explaination of results
F)    Exit
A
Generating 10 more recommendations...
|    | artist_name        | track_name
|---:|:-------------------|:------------------------------------
| 10 | Mac Powell         | God of Wonders
| 11 | Jamie Grace        | Hold Me (feat. Tobymac)
| 12 | Kelly Minter       | How Deep the Father's Love - Liv
| 13 | Tree63             | Blessed Be Your Name
| 14 | Wayfarer           | Alas and Did My Saviour Bleed?
| 15 | Chris Tomlin       | Everlasting God - See The Mornin
| 16 | Hillsong Chapel    | Hosanna - Live
| 17 | Kari Jobe          | You Are for Me
| 18 | Passion            | You Never Let Go - Live
| 19 | Natalie Grant      | Greatness Of Our God
```

Fig. 3. Interface of our recommender system. On the left shows the users current playlist and the generation of recommendations on the right.

V. Evaluation

As we can see in our interface examples in fig 3., our recommender system tends to give good results. For the user's current playlist, we can see the playlist genre leans towards Christian music. From simple observations of titles, we can see from our outputted recommendations, the songs predicted, seem to match well. However, the best way to evaluate our whole system would be to use evaluation metrics over all playlists rather than just observing.

For our evaluation methods we will be comparing our personalized (CNCF) and our baseline (Popularity Based) recommender system. We will be looking at evaluation metrics, one for rating accuracy, RMSE and two for ranking accuracy, Hit Ratio and NDCG using our train-test split from our previously discussed leave one out method. An issue with evaluating this recommender system is to evaluate the clustering as well as the NCF model. We give two different evaluations, one just from the score averages on evaluating solely our NCF models; the other by predicting the cluster of the song based on our Kmeans model and if it is correct, continue to our NCF evaluation otherwise, give it an automatic 0.

As we have a lack of evidence for measuring rating accuracy for our dataset and model and with evidence of similar circumstance evaluation shown in [19], we decided to use RMSE (11) as it also heavily focuses on large errors and aims to reduce the difference between our predicted and actual ratings which is beneficial for with our dataset. As we are working with percentage predictions our RMSE will be between 0-1, the lower the better.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{n}} \tag{11}$$

where y_i, $\hat{y}$, and n are predicted value, actual value and total number of values

Hits Ratio (HR) (12) and Normalized Discounted Cumulative Gain (NDCG) (13) are popular ranking evaluation metrics for evaluating the ranking of our recommendations for NCF as found in [20]. We will use these as they are able to evaluate large datasets and are normalized to give outputs between 0 and 1, so we can easily compare different recommender systems. They are also the best for our leave one out method test set. Our HR focuses on how relevant our song is to our user by measuring how many times our test track appears in the top K recommendations. On the other hand, NDCG focuses on the ranking position of our test track and gives a higher score the higher ranked the track. The results of our predictions are outputted as percentages of how likely they are to be interacted with [16] in the rankings. As it is too time-consuming to rank all the items for each user, we use a sample of 99 random tracks that are not interacted with by the user along and our left-out track [11] and rank these for every playlist. We will be following [15] where K = 10 is used is as the best value for our evaluation metric. The output is a percentage, the higher the better.

$$HR = \frac{hits}{hits + misses} = 1 - MR \tag{11}$$

$$nDCG_{pos} = \frac{DCG}{IDCG} \tag{12}$$

where $DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{log_2 i}$ $IDCG_{pos} = rel_1 + \sum_{i=2}^{|h|-1} \frac{rel_i}{log_2 i}$ *,rel_i is the graded relevance of the result at position i, and |h| is the list of relevant items up to the position i.*

TABLE I.

Evaluation Scores	***RMSE***	***HR@10***	***NDCG@10***
Personalised	0.5546	0.5810	0.4027
Personalised w Kmeans	0.9083	0.3175	0.1628
Non-Personalised	0.9985	0.2278	0.0368

Fig. 4. Table summarising the evaluations metrics on recommender systems

From our results in fig. 4 we can see that our personalised recommender has done better than our popularity-based RS. Our first row shows the results of the average evaluation scores on all the NCF models alone. However, our second row shows the evaluation, whilst considering our previous Kmeans clustering. For both personalised evaluations, our RMSE has a lower score whilst our Hits@10 and NDCG has a higher score than our popularity-based RS. As our model puts an emphasis on user and item similarity and models 65464776 tracks and 1 million playlists, we are not surprised that it has performed better than our baseline model. However, we can see that when including an evaluation on Kmeans that our all our evaluation scores are worse than without evaluating with Kmeans. This shows that there is a possibility that our scores could be much better if we improved our clustering technique. Despite this, we would most definitely agree that our Clustering Neural Collaborative Filtering on the MPD is beneficial and can be used with more research.

VI. Conclusion

For this project we have investigated Clustering Neural Collaborative Filtering is for a music recommender system using the Million Playlist Dataset. During our process, we cleaned our data by removing irrelevant data and created a dataset based on songs. We also used a Spotify API to collect audio features for each playlist. Using our hybrid of clustering and neural collaborative filtering we proved that it helps the scalability issue of our data set and how content features have helped combat the cold start problem. When comparing it to our baseline popularity recommender system we found that it yielded a better ranking and rating accuracy overall.

A. *Limitations*

One of our limitations is that we need to have trained our playlist to recommend tracks as we need to use its embeddings. Therefore, if we were to create a new playlist, we would need to retrain our model which is not effective in terms of time, computing costs and adaptability. Another limitation is that the clustering algorithm is that it may not be able to accurately group songs together. This can lead to poor performance on our NCF algorithm as it is making recommendations based on inaccurate clusters. In addition, clustering-based methods are sensitive to the number of clusters chosen and the initialization of the centroids [14], which can make our choice of 100 clusters premature and ineffective overall.

B. *Future Work*

For future improvements it would be good to investigate into using our track audio features in our neural collaborative system found by [21]. Another implementation can be to use the Spotify API for the user to be able to input their playlist from their own Spotify account. Also, to look into how we can add a new user efficiently in our system, if we can't adapt our current system, can we add cosine similarity to map it to a similar playlist?, if we can't adapt our current system, can we add cosine similarity to map it to a similar playlist?

References

[1] C.W. Chen, P. Lamere, M. Schedl, and H. Zamani. Recsys Challenge 2018: Automatic Music Playlist Continuation. In Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18), 2018.

[2] The Clever Programmer, "Spotify Recommendation System with Machine Learning" https://thecleverprogrammer.com/2021/03/03/spotify-recommendation-system-with-machine-learning/ (Accessed Jan 3 2023)

[3] Serhii Ripenko, Nadiia Hasiuk "Music Recoomendation system: all you need to know" https://www.eliftech.com/insights/all-you-need-to-know-about-a-music-recommendation-system-with-a-step-by-step-guide-to-creating-it/ (Accessed Jan 3 2023)

[4] Ferraro, A. et al. (2021) 'Melon Playlist Dataset: A Public Dataset for Audio-Based Playlist Generation and Music Tagging', in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE, pp. 536–540. doi:10.1109/ICASSP39728.2021.9413552.

[5] Paul Lamere, Spotipy, (2020), A light weight Python library for the Spotify Web API, https://github.com/plamere/spotipy

[6] Chen, C.-W. et al. (2018) 'Recsys challenge 2018: automatic music playlist continuation', in Proceedings of the 12th ACM Conference on recommender systems. ACM, pp. 527–528. doi:10.1145/3240323.3240342.

[7] Yanru Jiang and Xin Jin. 2022. Using k-Means Clustering to Classify Protest Songs Based on Conceptual and Descriptive Audio Features. In Culture and Computing: 10th International Conference, C&C 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26 – July 1, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, 291–304. https://doi.org/10.1007/978-3-031-05434-1_19

[8] Animesh Goyal "Solving Cold User problem for Recommendation system using Multi-Armed Bandit" https://towardsdatascience.com/solving-cold-user-problem-for-recommendation-system-using-multi-armed-bandit-d36e42fe8d44 (Accessed Nov 15 2023)

[9] He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.S., 2017, April. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

[10] Matt Payne "Neural Collaborative Filtering for Deep Learning Based Recommendation Systems | Architecture Breakdown & Business Use Case" https://www.width.ai/post/neural-collaborative-filtering (Accessed Jan 08 2023)

[11] Vainavi Samant "Deep Learning based Recommender Systems" https://www.kaggle.com/code/vainavisamant/deep-learning-based-recommender-systems#Evaluating-our-Recommender-System (Accessed Jan 08 2023)

[12] Abba Suganda Girsang et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1071 012021

[13] Chen, J., Wang, B., Ouyang, Z. et al. Dynamic clustering collaborative filtering recommendation algorithm based on double-layer network. Int. J. Mach. Learn. & Cyber. 12, 1097–1113 (2021).

[14] Soner Yildirim "Two Challenges of K-Means Clustering" https://towardsdatascience.com/two-challenges-of-k-means-clustering-72e90bdeb0da (Accessed Jan 10 2023)

[15] Bhaskar, K.R.K., Kundur, D. and Lawryshyn, Y. (2020) 'Implicit Feedback Deep Collaborative Filtering Product Recommendation System'. doi:10.48550/arxiv.2009.08950.

[16] Cara Vanuden "Neural Collaborative Filtering with NeuMF" https://caravanuden.com/spotify_recsys/neural_collaborative_filtering.html (Accessed Dec 30 2022)

[17] Abhishek Sharma "Neural Collaborative Filtering" https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401 (Accessed Jan 2 2023)

[18] Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., & Elahi, M. (2018). Current challenges and visions in music recommender systems research. International Journal of Multimedia Information Retrieval, 7(2), 95–116. https://doi.org/10.1007/s13735-018-0154-2

[19] Leah Ajmani, Chen Hu, and Ruixuan Sun. 2021. Replication Assignment. 1, 1 (February 2021), 4 pages.

[20] 'Neural collaborative filtering vs. matrix factorization revisited' (no date) in. doi:10.1145/3383313.3412488.

[21] van den Oord, A., Dieleman, S. and Schrauwen, B. (2013) 'Deep content-based music recommendation', in. Neural Information Processing Systems Foundation (NIPS)