Published in AJSE, Vol:22, Issue: 2
Received on 11th August 2022
Revised on 22nd February 2023
Accepted on 30th June 2023

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 – 3679 (print) 2520 – 4890 (Online)

A Recommendation System Based on Implicit Data for Internet Protocol Television (IPTV)

Lama Mohsen Mansour, Zainab Omran, Ghaydaa Kaddoura,
Mustapha Dakkak and Yasser Rahhal

***Abstract*--** **IPTV delivers television content over Internet Protocol (IP) networks. Videos On Demand (VOD) is the most popular IPTV, allowing users to freely select from a vast pool of program genres. Therefore, it is necessary to introduce innovative features to attract new users and retain existing ones. For this purpose, IPTV systems typically use VOD recommendation engines. The primary purpose of recommendation systems is to suggest user-relevant items from various items by producing a list of recommendations for each user. In this paper, we introduce an approach to recommendation systems in IPTV. We developed this approach on implicit feedback derived from users' interaction with movies/series sets, such as how many times they watched a movie and how long they have spent watching specific movies/series. For the previous factors, we tested a variety of recommendation algorithms, content-based, collaborative-based, and hybrid. Then applied the previously mentioned algorithms on real-life big data sets after introducing some modifications to the algorithms, then benchmarked the results on multiple performance metrics. We noticed that the applied changes achieved promising results.**

Keywords: Recommendation system (RS), implicit data, Big Data, Performance Metrics, Videos On Demand (VOD), Internet Protocol Television (IPTV).

I. Introduction

Nowadays, IPTV offers an extensive range of movies content. On the one hand, the more channels we have, the higher the chance of each viewer finding movies or series of their preference; on the other hand, it poses severe challenges in navigating through the program grid [1]. A common approach to building such a user preference model is either explicitly or implicitly eliciting user feedback. Explicit feedback, such as rating scales, provides a direct mechanism for users to express their interests in items. The RS itself generates implicit feedback using the interpretation it makes about the user's behavior. What constitutes implicit feedback depends on the application domain. Usually, it will be one or multiple observable and measurable parameters that arise from the user's interactions with the RS [2].

There are majorly three categories of RSs: Content-based, Collaborative, and Hybrid approaches, a content-based RS analyses the user's past behavior and keeps track of patterns to predict and suggest items that match these patterns [3]. Suggestions are calculated based on characteristics of the item, such as category, actor, etc. Collaborative filtering suggests items based on the preferences of users with similar tastes and interests using correlation-based similarity [4] or generative model [5], clustering [6], Matrix factorization [7] [8], and deep learning [9] [10]. Both content-based and collaborative-based filtering have their drawbacks. To avoid this. Researchers suggested a hybrid approach to combine two or more recommendation strategies in different ways to benefit from their complementary advantages [2]. This paper presents various recommendation methods and compares them based on distinct metrics to highlight multiple options according to the application need. Moreover, it is imperative to offer innovative features to attract new users and retain existing ones. In practice, a good recommendation engine does not offer popular and well-known titles. Still, it can instead identify compelling titles among less popular items that would otherwise be hard to find [11] [12].
Furthermore, to evaluate the recommendation model, it is essential to know how many items are suggested [13].

II. Related Work

Two different directions of RSs have evolved, content-based filtering and collaborative filtering. Content-based RSs typically use movie features and users' viewing profiles to recommend items similar to the items the user has interacted with before. Researchers in [14] introduced a method to deal with implicit feedback and build a content-based movie RS that can use different feature sets: name, actor, director, category, and writer features. They established a feature set based on the particular user's past behavior and assigned a weight for each feature. They also produce a user's implicit rating for a movie based on the movie's duration that the user viewed. Furthermore, to predict a movie rating, they merge the user-specific weights of the movie's features using a particular feature set. We follow the previous approach with a few modifications in the rating prediction method to make it more logical. We reevaluate the results with different evaluation methods, thoroughly discussed in the next section. We typically obtained collaborative filtering predictions about user interests by grouping users with similar tastes [15]. We use collaborative filtering methods such as the

Lama Mohsen Mansour,
Higher Institute for Applied Sciences and Technology, Damascus, Syria
Email : lama.mohsen.mansour@gmail.com

Zainab Omran,
Higher Institute for Applied Sciences and Technology, Damascus, Syria

Ghaydaa Kaddoura
Higher Institute for Applied Sciences and Technology, Damascus, Syria

Mustapha Dakkak
Higher Institute for Applied Sciences and Technology, Damascus, Syria

Yasser Rahhal
Higher Institute for Applied Sciences and Technology, Damascus, Syria

nearst neighbor (NNH) algorithm combined with the Pearson correlation or cosine similarity to calculate the predicted values. In terms of sparse rating, the NNH approach usually experiences difficulties in finding the right match. In addition, the algorithm complexity tends to increase with the number of users and the number of items [15]. To overcome these difficulties, we adopted matrix factorization such as ALS, which projects users and items into a shared latent space using techniques such as Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), and others [16]. It represents a user or an item by a vector of latent features, and a user's interaction with an item is modeled as the inner product of their latent vectors [17]. These techniques (collaborative-Based and Content-Based) have their setbacks, such as the cold-start problems in collaborative filtering and content-based filtering. We can try many hybrid techniques used in RSs to increase recommendation accuracy and reduce errors to deal with these setbacks. The most used algorithm is the NNH combined with the matrix factorization [15]. Therefore, we depended on the combination between ALS and NNH using the rating built-in [14] and designed the parallel implementation process of the recommendation algorithm
based on the Spark platform [18]. However, the attention is increasingly shifting towards implicit data and creating a model that can predict the score of unobservable items from the (user, item) interaction matrix, which can be filled with 1 if the user u interaction with the item i is observed and 0 otherwise. We followed the approach used in[17] in which the researchers found that the Neural Collaborative Filtering (NCF) framework offers better recommendation performance than Matrix factorization. However, we found another perspective if the objective of the RS is to recommend unexpected items [11]. We know that an excellent recommender system makes both relevant and valuable recommendations; many papers focus on understanding the performance of RSs. In [19], the researchers compared standard recommendation algorithms using Root Mean Square Error (RMSE), Normalized Discounted Cumulative Gain (NDCG), and item coverage. In [20], the researchers assessed evaluation metrics such as recall, F-measure, accuracy, novelty, and Global satisfaction on various recommendation methods. Our research differs from previous works in that it involves six evaluation methods applied to four different recommendation algorithms.

III. Experiments

3.1 Dataset

We experimented with a real dataset extracted from the IPTV application called ISHOW that contains multiple items, including movies and series. Our experiments used 11 months of log data, the first 10 months for the training, and the last month for the test phase. The characteristics of the dataset are summarized in Table 1 and Table 2.

Table 1. Statistics of the training dataset

Dataset name	**Interaction #**	**Item #**	**User #**	**Sparsity**
Movies	829314	12864	64376	99.89%
Series	738365	1576	87549	99.46%

Table 2. Statistics of the testing dataset

Dataset name	**Interaction#**	**Item#**	**User#**
Movies	108064	11795	8722
Series	143987	1199	28125

In Fig. 1, we noted the number of users according to watched movies and series.

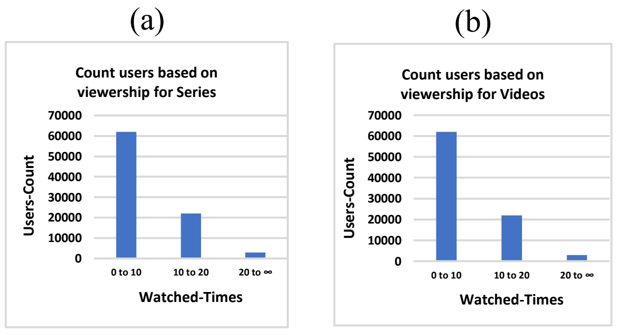


Fig. 1: The count of interactive users according to the count of watched series then movies

3.2 Building Several Types of Recommender Systems

We experimented with various RS algorithms with some new modifications, and then we defined the evaluation protocol. Finally, we analyzed the results.

3.2.1 Rating calculation based on content-based RS using feature sets called an acronym CB

We repeated the same approach in [5], where the researchers generated the rating by summarizing the feature weights of the contents from the user's perspective but based on questionnaires, we found that the category feature is the most important. So, we gave it more weight i ∈{1, 1.5, 2, 2.5} as in Equation (1).

$$Rate\ (U,S) = \sum_{F=1\ \&\ F!=cat}^{NF} W(U,F)\left(\sum_{F=cat}^{CAT} W(U,F)\right. \quad \dots (1)$$

Where, U indicates to the user, S indicates to item, cat indicates the category, NF is the feature set for each item, whose feature set ∊ {heroes, directors, writers, categories}, and W (U, F) indicates the weight of feature F in the feature set NF.
Table 3 shows the recommendation performance (recall, precision) for movies and series.

Table 3. The recommendation performance

	Movies		Series	
Weight	**Precision**	**Recall**	**Precision**	**Recall**
i=1	0.149	0.09	0.31	0.38
i=1.5	0.156	0.24	0.33	0.91
i=2	0.169	0.63	0.32	0.18
i=2.5	0.161	0.37	0.29	0.09

We noticed that the recommendation performance is better when i=2 for movies, but the series is better when i=1

3.2.3 *Combination of matrix factorization and classical collaborative filtering (nearest neighbor) called an acronym MF&KNN*

Matrix factorization is the most recent solution for sparse data problems, although it has become widely known since Netflix Prize Challenge [21]. Alternating Least Square (ALS) is a matrix factorization algorithm, and it runs itself in a parallel way. ALS is implemented in Apache Spark ML and built for a large-scale collaborative filtering problem. ALS is doing a pretty good job at solving the scalability and sparseness of the rating data, and it's simple and scales well to massive datasets.
We adopted the approach in [15]. So, we tried the ALS approach used in spark. mllib. Our workflow is following [2], [22]:

i. We load extracted rating data from the previous method rating calculation based on content using feature sets.
ii. To reduce the effect of outliers, we removed all users that have only a single or two actions. They will not help us learn any meaningful relationships, though, which is what we want.
iii. Build the recommendation model using ALS that is built-in spark 2.3 on the training data 1.
iv. Generate top 10/50 movie recommendations for each user.
v. For users who have only a single or two actions, we can instead run a KNN search over the product matrix to find movies/series most similar to those of users.

3.2.3 *Neural Collaborative Filtering based on actions called an acronym NCF-A*

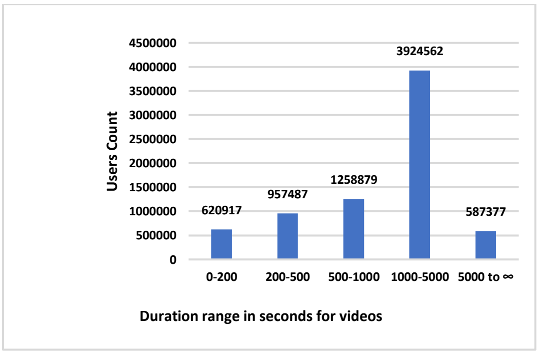
We applied the NCF method as mentioned in [17], we filled the (user, item) interaction matrix with 1 if the user u's interaction with the item i is observed, and 0 otherwise, we initiated the training set for every item that the user watched, we added four negative items the user hasn't watched randomly, regarding testing set for every user we added 100 items, 99 one of them, the user has never watched it before, and one is the last item the user watched to measure hit ratio (HR), and normalized discounted cumulative gain (NDCG) metrics.
Users who have watched only one item will be excluded from the training and testing set.

3.2.4 *Neural Collaborative Filtering based on duration called an acronym NCF-D*

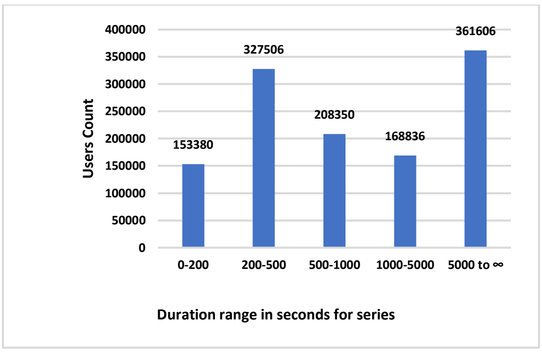
We added some modifications to the NCF method making use of the time that the user spent on an item as an indirect measure of his feedback based on a specific threshold (500 seconds for movies and 1000 seconds for series), we filled the (user, item) interaction matrix with 1 if the watched duration of the user u to the item i is more than it, and 0 otherwise. Fig. 2 represents the duration distribution based on the number of watched movies and series.
To deal with the negative feedback, we follow the following steps:

- For users with zero negative items, we added twice the count of positive items randomly.
- For users whose count of positive items less than twice the negative items count, we did the following strategy:

i. We used the Count Vectorizer feature extraction method from the scikit-learn library to turn every item into a feature vector using the bag of words from (genre, writers, directors, actors), and we used the cosine similarity to measure the similarity between two items.
ii. We added negative items with the count S = (2× positive count items – negative count items) by taking the top S items most similar to the negative items in the training set.
iii. For the users whose items we didn't find similar ones, we added S items randomly.



(a)



(b)

Fig. 2 The duration distribution based on the number of watched movies then series

3.3 Evaluation Protocols

This section investigates our experiments using multiple evaluation dimensions, covering coverage, novelty, diversity, NDCG, HR, and percentage of correctly captured predictions.

3.3.1 HR and NDCG

We adopted the leave-one-out procedure, which has been widely used in the studies [23] [24] [25]. We excluded the last watched item for each user and checked whether it's included in the predicted top 10 list. We call this the Hit Ratio (HR) metric. We also calculated the item's location in the predicted list as a normalized discounted cumulative gain (NDCG) metric [26]. As for the remaining items in the watched list, we use them as a training set [10].

Table 4. Evaluation metric results 1

	Movies		Series	
Recommendation method	**HR**	**NDCG**	**HR**	**NDCG**
CB	0.36	0.21	0.29	0.17
MF&KNN	0.25	0.12	0.12	0.06
NCF-A	0.85	0.66	0.80	0.51
NCF-D (1,0)	(0.66, 0.44)	(0.44, 0.24)	(0.60, 0.30)	(0.22, 0.13)

Table 4 shows (HR & NDCG) for the four-recommendation method. In the NCF-D method, we will symbolize the last watched item with 1 over the threshold and 0 otherwise.

3.3.2 Percentage of correctly captured predictions

It represents the percentage of the recommendations, which has at least a given number of successful recommendations

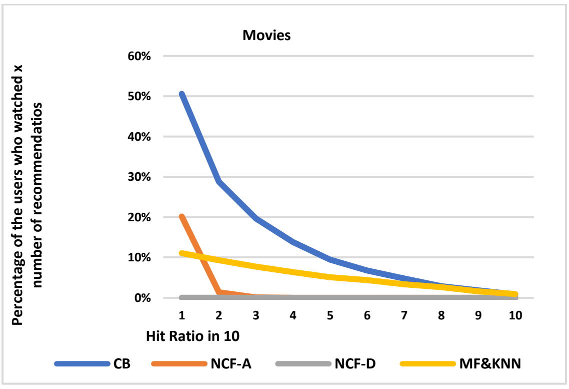
This metric is generally similar to the HR approach, the main difference is that we maintained all elements of training and after recommending each user's top 10 list, we calculated the percentage of the recommendations that appeared in the user's watched list [14].

Table 5 shows the mean of successful recommendations for each recommendation method called an acronym Mean- PCCP.

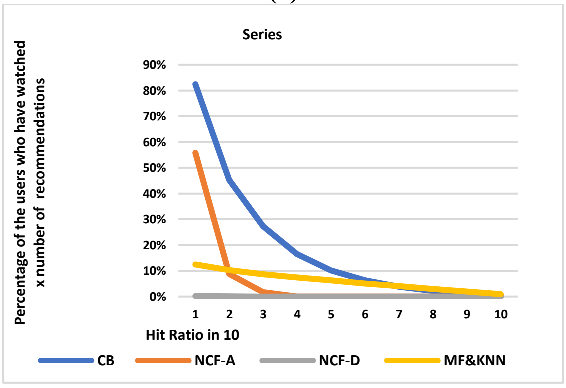
Table 5. Evaluation metric results 2

	Mean- PCCP	
Recommendation method	**Movies**	**Series**
CB	13.96	19.50
NCF-A	2.17	6.63
MF&KNN	5.24	6.01
NCF-D	0.003	0.05

Fig. 3 represents the successful recommendations counts for each algorithm on the two datasets.



(a)



(b)

Fig. 3 Hit counts for each algorithm on the two datasets

3.3.3 Diversity measures

Most researches aimed mainly on enhancing the efficiency of recommender systems and relatively on developing the utility of the recommendation list. Therefore, we assessed the diversity of items on the recommendation list.

We adopted the definition of set diversity [11] to model diversity as the aggregate, or, identically, an average difference of all pairs of items in the set. Specifically, given a distance function, d: I×I→R, such that d (i, j) is the distance or difference between elements i, j ∈ I, the diversity f_D (R) is the average dissimilarity of all pairs of elements contained in R as in Equation (2).

$$\boldsymbol{f_D(R)} = \frac{\mathbf{1}}{\boldsymbol{p(p-1)}} \sum_{i \in R} \sum_{j \in R, j \neq i} \boldsymbol{d(i,j)} \qquad \textbf{… (2)}$$

Where p=|R|. Here we assume that this distance function is symmetric (d (i, j) =d (j, i)).

Table 6 shows the diversity value based on category, then all features for the four recommendation methods.

Table 6. Evaluation metric results 3

	Movies		**Series**	
Recommendation method	**Diversity based on Category only**	**Diversity based on all features**	**Diversity based on Category only**	**Diversity based on all features**
CB	0.51	0.83	0.22	0.51
NCF-A	0.75	0.90	0.67	0.88
MF&KNN	0.59	0.86	0.23	0.52
NCF-D	0.78	0.93	0.69	0.90

3.3.4 Coverage

Item coverage measures the proportion of items a recommender system can recommend from the entire item space [27].

Fig. 4 shows item coverage of recommendation lists produced by different recommender systems for series then movies.

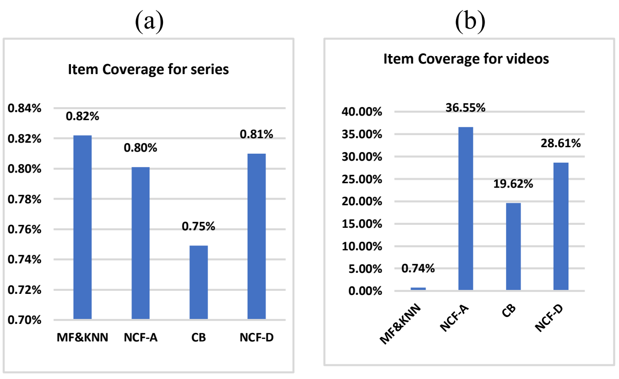


Fig. 4 Item coverage of recommendation lists produced by different recommender systems for series then movies.

3.3.5 Novelty

An item will be considered novel if it is difficult to find in a given dataset. The degree of difficulty in discovering a new item will depend on the ratings that an item receives [28]. In our case, we will rely on the number of times that an item has been watched. Wherever the viewership of recommended items is less (the item is watched once or never watched {0, 1}), it is better to be recommended.

Fig. 5 represents the percentage of recommended items viewership by total users for each recommended item in the top 10 for the series.

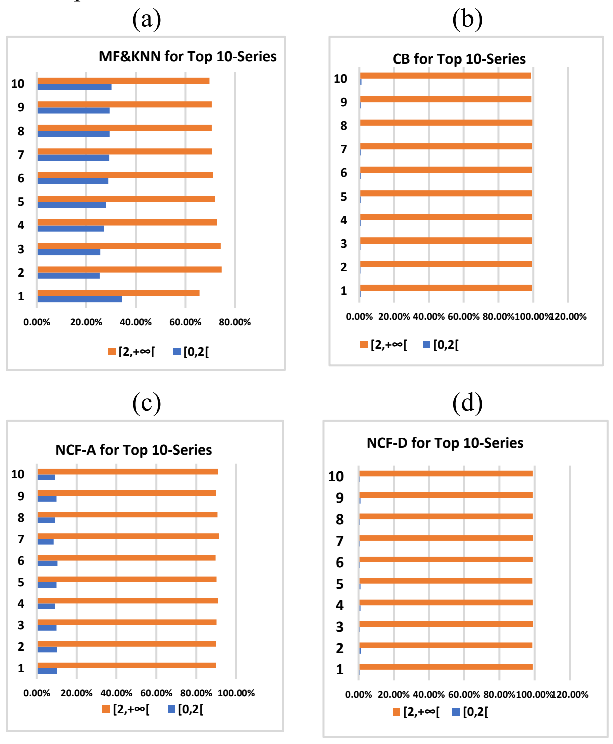
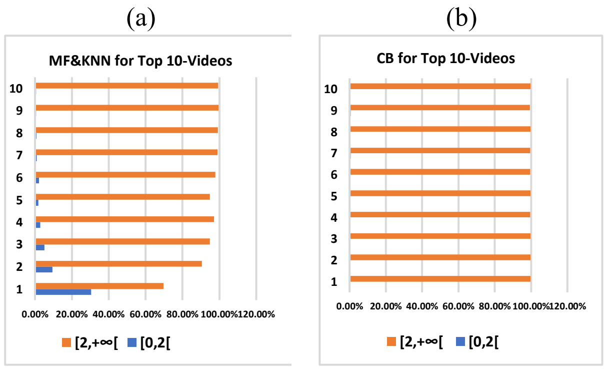
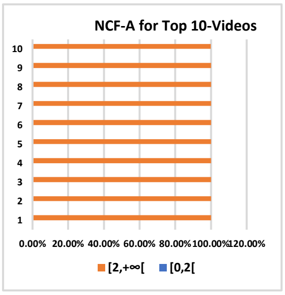


Fig. 5 The percentage of recommended items viewership ([0, 2[, [2, +∞ [) by total users for each recommended item in the top 10 for series by each recommendation method.

Fig. 6 represents the percentage of recommended items viewership by total users for each recommended item in the top 10 for movies.



(c)



(d)

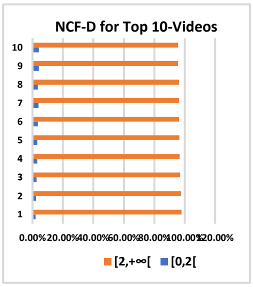


Fig. 6 represents the percentage of recommended items viewership ([0, 2[, [2, ∞ [) by total users for each recommended item in the top 10 for movies by each recommendation method.

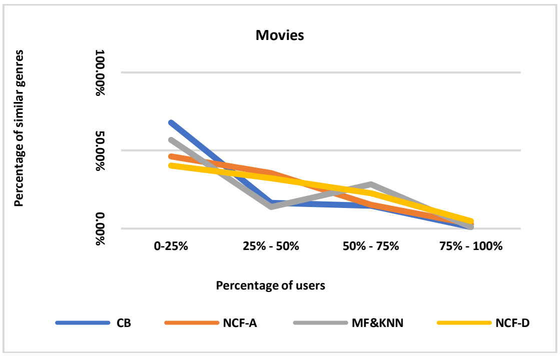
Table 7. Evaluation metric results 4

	Mean of novelty	
Recommendation method	**Movies**	**Series**
CB	0.42%	0.85%
NCF-A	0.03%	9.68%
MF&KNN	5.46%	28.78%
NCF-D	3.27%	1.03%

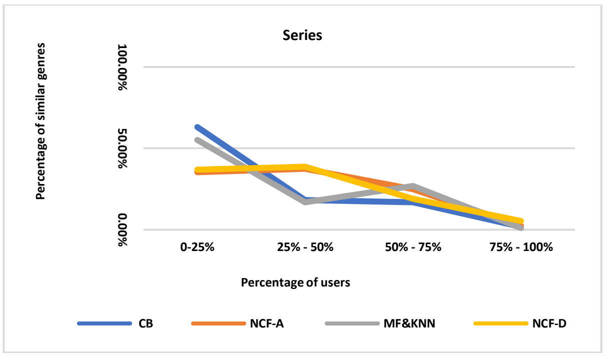
IV. Results and Discussions

Firstly as preliminary insight into results, we'll show the percentage of common categories for the users between their watched items during training phase and each recommended item in the top 10 list for movies then series by each recommendation method.

In Fig 7 we noticed that NCF-D method has the most common categories in recommendation methods for the percentage of users 75% - 100% and that for movies and series.



(a)



(b)

Fig. 7 represents the percentage of common genres for the users between their watched items during training phase and their ones during test phase for movies/series.

Although no algorithm is significantly better (or worse) than all the other in terms of quality dimensions, we can at least identify a partial order [20], as outlined in Table 8, Table 9 for series then movies.

Table 8. Partial Ordering of RSs for the various evaluation methods on series

	HR & NDCG	***Mean-PCCP***	**Coverge**	**Mean of novelty**	**Diversiy**
Maximal	*NCF-A*	*CB*	*MF&KNN*	*MF&KNN*	*NCF-D*
Upper-Intermediate	*NCF-D*	*NCF-A*	*NCF-D*	*NCF-D*	*NCF-A*
Intermediate	*CB*	*MF&KNN*	*NCF-A*	*NCF-A*	*MF&KNN*
Minimal	*MF&KNN*	*NCF-D*	*CB*	*CB*	*CB*

Table 9. Partial Ordering of RSs for the various evaluation methods on movies

	HR & NDCG	***Mean-PCCP***	**Coverge**	**Mean of novelty**	**Diversiy**
Maximal	*NCF-A*	*CB*	*NCF-A*	*MF&KNN*	*NCF-D*
Upper-Intermediate	*NCF-D*	*MF&KNN*	*NCF-D*	*NCF-D*	*NCF-A*
Intermediate	*CB*	*NCF-A*	*CB*	*CB*	*MF&KNN*
Minimal	*MF&KNN*	*NCF-D*	*MF&KNN*	*NCF-A*	*CB*

According to this ordering, *NCF-A* is the most optimal algorithm in terms of relevance (i.e., the algorithm with the best-perceived relevance) for two datasets. On the contrary, MF&KNN is the least optimal algorithm for them.

The second column of the table shows the partial ordering according to Mean-PCCP. CB is the algorithm that mostly matched users' interests in general for the two datasets, while MF&KNN showed fewer matches to users' interests in them.

The third column of the table shows the partial ordering according to the coverage. MF&KNN is the most optimal algorithm for the series. However, the most optimal algorithm for movies is NCF-A, whereas the least optimal algorithm for series and movies are CB and MF&KNN, respectively.

The fourth column of the table shows the partial ordering according to the novelty. MF&KNN is the most optimal algorithm for two datasets, while the least optimal algorithm for series and movies are CB and NCF-A, respectively.

The last column of the table shows the partial ordering according to the diversity. *NCF-D* is the maximal algorithm for the two datasets, while *CB* is the minimal algorithm.

V. Conclusions

In general, we choose the recommendation method according to the application context. This paper experimented with four different types of RS methods to highlight the best use-cases for each method. Furthermore, we presented their differences based on multiple factors, adopting an offline evaluation approach. Therefore, we suggest using an online evaluation approach to test the RSs we previously mentioned and re-analyze the results in future studies.

Acknowledgement

Syriatel telecom Co-sponsored this research. We thank our colleagues Mr. Murid (Syriatel CEO), Mr. Ayman (Syriatel CTO), Mr. Adham (Research & Development Manager), who provided insight that aided the research. In addition, we thank Wessam Abo Zayed for his great ideas in the data processing.

References

[1] E. Bonilla, M. Stier, S. Niccolini, and M. Brunner, “Automated Real-Time Recommendations for IPTV,” Nov. 2009, pp. 1–6. doi: 10.1109/icc.2009.5305951.

[2] G. Jawaheer, M. Szomszor, and P. Kostkova, “Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service,” in Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, 2010, pp. 47–51. doi: 10.1145/1869446.1869453.

[3] Peter. Brusilovsky, Association for Computing Machinery., and S. ACM Conference on Recommender Systems (4th : 2010 : Barcelona, Content-based movie recommendation system using genre correlation. In Smart Intelligent Computing and Applications. Association for Computing Machinery, 2010.

[4] R. H. Singh, S. Maurya, T. Tripathi, T. Narula, and G. Srivastav, “Movie Recommendation System using Cosine Similarity and KNN,” International Journal of Engineering and Advanced Technology (IJEAT), no. 9, pp. 2249–8958, 2020, doi: 10.35940/ijeat.E9666.069520.

[5] S. Biswas, L. V. S. Lakshmanan, and S. B. Roy, “Combating the Cold Start User Problem in Model Based Collaborative Filtering,” ArXiv, vol. abs/1703.00397, 2017.

[6] N. F. AL-Bakri and S. H. Hashim, “Collaborative Filtering Recommendation Model Based on k-means Clustering,” Al-Nahrain Journal of Science, vol. 22, no. 1, pp. 74–79, Mar. 2019, doi: 10.22401/ANJS.22.1.10.

[7] Y. He, C. Wang, and C. Jiang, “Correlated Matrix Factorization for Recommendation with Implicit Feedback,” IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 3, pp. 451–464, Mar. 2019, doi: 10.1109/TKDE.2018.2840993.

[8] M. Patil, S. Brid, and S. Dhebar, “COMPARISON OF DIFFERENT MUSIC RECOMMENDATION SYSTEM ALGORITHMS”.

[9] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, “A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System,” IEEE Transactions on Cybernetics, vol. 49, no. 3, pp. 1084–1096, 2019, doi: 10.1109/TCYB.2018.2795041.

[10] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, “Neural Collaborative Filtering vs. Matrix Factorization Revisited,” in RecSys 2020 - 14th ACM Conference on Recommender Systems, Sep. 2020, pp. 240–248. doi: 10.1145/3383313.3412488.

[11] N. Hurley and M. Zhang, “Novelty and Diversity in top-N recommendation-Analysis and evaluation,” ACM Transactions on Internet Technology, vol. 10, no. 4, Mar. 2011, doi: 10.1145/1944339.1944341.

[12] K. Pripužić et al., “Building an IPTV VoD recommender system: An experience report DL-Tags: DLT and Smart Tags for decentralized, privacy-preserving and verifiable supply chain management View project Building an IPTV VoD Recommender System: An Experience Report,” 2013. [Online]. Available: https://www.researchgate.net/publication/256437748

[13] X. Yin et al., “Time Context-Aware IPTV Program Recommendation Based on Tensor Learning,” in 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1–6. doi: 10.1109/GLOCOM.2018.8647211.

[14] M. Uluyagmur, Z. Cataltepe, and E. Tayfur, “Content-based movie recommendation using different feature sets,” in Proceedings of the world congress on engineering and computer science, 2012, vol. 1, pp. 17–24.

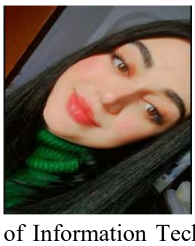
[15] M. Ilhami and Suharjito, “Film recommendation systems using matrix factorization and collaborative filtering,” in 2014 International Conference on Information Technology Systems and Innovation, ICITSI 2014 - Proceedings, Feb. 2014, pp. 1–6. doi: 10.1109/ICITSI.2014.7048228.

[16] Y. Koren, R. Bell, and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” Computer, vol. 42, no. 8, pp. 30–37, 2009, doi: 10.1109/MC.2009.263.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, “Neural collaborative filtering,” in 26th International World Wide Web Conference, WWW 2017, 2017, pp. 173–182. doi: 10.1145/3038912.3052569.

[18] B. Kupisz and O. Unold, “Collaborative filtering recommendation algorithm based on Hadoop and Spark,” in

2015 IEEE International Conference on Industrial Technology (ICIT), 2015, pp. 1510–1514.
[19] A. Said and A. Bellogín, "Comparative recommender system evaluation: Benchmarking recommendation frameworks," in RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems, Oct. 2014, pp. 129–136. doi: 10.1145/2645710.2645746.
[20] P. Cremonesi, F. Garzotto, S. Negro, A. V. Papadopoulos, and R. Turrin, "LNCS 6948 - Looking for 'Good' Recommendations: A Comparative Evaluation of Recommender Systems," 2011.
[21] Y. Koren, "The BellKor Solution to the Netflix Grand Prize," 2009. [Online]. Available: www.netflixprize.com/leaderboard
[22] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," in Smart Innovation, Systems and Technologies, 2019, vol. 105, pp. 391–397. doi: 10.1007/978-981-13-1927-3_42.
[23] S. A. Stein, G. Weiss, Y. Chen, and D. D. Leeds, "A College Major Recommendation System," Fourteenth ACM Conference on Recommender Systems, 2020.
[24] G. Stamatelatos, G. Drosatos, S. Gyftopoulos, H. Briola, and P. S. Efraimidis, "Point-of-interest lists and their potential in recommendation systems," Information Technology and Tourism, vol. 23, no. 2, pp. 209–239, Jun. 2021, doi: 10.1007/s40558-021-00195-5.
[25] O. Jeunen, "Revisiting offline evaluation for implicit-feedback recommender systems," in RecSys 2019 - 13th ACM Conference on Recommender Systems, Sep. 2019, pp. 596–600. doi: 10.1145/3298689.3347069.
[26] H. Xiao, Y. Chen, and X. Shi, "Multi-Perspective Neural Architecture for Recommendation System," Jul. 2018, [Online]. Available: http://arxiv.org/abs/1807.09751
[27] Y. Chung, N. R. Kim, C. Y. Park, and J. H. Lee, "Improved neighborhood search for collaborative filtering," International Journal of Fuzzy Logic and Intelligent Systems, vol. 18, no. 1, pp. 29–40, Mar. 2018, doi: 10.5391/IJFIS.2018.18.1.29.
[28] M. Mendoza and N. Torres, "Evaluating content novelty in recommender systems," Journal of Intelligent Information Systems, vol. 54, no. 2, pp. 297–316, Apr. 2020, doi: 10.1007/s10844-019-00548-x.



Lama Mohsen Mansour received her master degree in Decision Support Sytstem (DSS) from Higher Institute for Applied Sciences and Technology (HIAST) in 2022. She has served as Aritificial Intellegence Reasearch and Development Supervior as Syriatel since 2018. She finished her Bachelor of Information Technology (IT) in Softwaring Engeneering from Albaath unversity.

Zainab Omran was born in Damascus, Syria in 1995. She attended Damascus University, where she pursued a degree in Information Technology. After several years of hard work and dedication, Zainab graduated in 2018 with a strong foundation in Artifical Intelligence and programming. Following graduation, Zainab landed a job as a data scientist at a telecommunications company. In 2020, Zainab began pursuing a master's degree in Artifical Intelligence at Damascus University, further expanding her knowledge and skills in the field. Outside Work and education, Zainab enjoys spending time exploring new technologies. With a passion for learning and a drive to succeed, Zainab is poised to make a significant impact in the field of data science in the years to come.

Ghaydaa Mnsoor Kaddoura received her bachelor degree of Information technology majoring in Artificial intelligence from Damascus University in 2019. She has served as a Data Scientist followed by R&D AI specialist at Syriatel Telecom from 2020 to 2022, she then served as a Data engineer at Ligadata from 2022 to 2023 and is now preparing to embark on a new journey as a Software Engineer at freiheit.com technologies.