Journal of Ambient Intelligence and Humanized Computing (2023) 14:16025–16036
https://doi.org/10.1007/s12652-022-04375-x

ORIGINAL RESEARCH

Remembering past and predicting future: a hybrid recurrent neural network based recommender system

Saumya Bansal[1] · Niyati Baliyan[1]

Received: 28 July 2021 / Accepted: 30 July 2022 / Published online: 4 September 2022

Abstract
Traditional recommender systems (RS) assume users' taste to be static (taste remains same over time) and reactive (a change in taste cannot be predicted and is observed only after it occurs). Further, traditional RS restricts the recommendation process to candidate items generation. This work aims to explore two phases of RS, i.e., Candidate Generation as well as Candidate Ranking. We propose a RS from a multi-objective (short-term prediction, long-term prediction, diversity, and popularity bias) perspective which was previously overlooked. The sequential and non-sequential behavior of users is exploited to predict future behavioral trajectories with the consideration of short-term and long-term prediction using recurrent neural networks and nearest neighbors approach. Further, a novel candidate ranking method is introduced to prevent users from being entangled in recommended items. On multiple datasets, largest being MovieLens (ML) 1M, our model shows excellent results achieving a hit rate and short-term prediction success of 58% and 71% respectively on ML 1M. Further, it implicitly handles two important parameters, i.e., diversity and item popularity with a success rate of 59.22% and 34.28% respectively.

Keywords Recommender systems · Nearest neighbors · Non-sequential recommendations · Recurrent neural network · Sequential recommendations

1 Introduction

A flurry of research works in the field of Recommender Systems (RS) came after the introduction of the Netflix challenge (Jannach et al. 2010; Bennett et al. 2007; Bansal and Baliyan 2019; Mardukhi et al. 2021) which has been further given a push by rise in the OTT usage since Covid-19. Unlike most businesses, the Covid-19 was game-changing for the entertainment and media industry. The year 2020 turned out to be a boom for OTT entertainment with movie theatres being shut down. Due to Covid-19, India's video streaming industry is all set to reach Rs. 11,977, growing at a CAGR of 21.82% by 2023.[1] Growth in the consumption of OTT with the existence of big players like YouTube, Amazon Prime Video, and Netflix, demands more research for personalized recommendations to save user's time in finding movie of interest and thus increase user's experience. For instance, to watch a movie on Netflix, a user might have to go through a large number of trailers before finding a movie of interest, which is a time-consuming process and may even end up not watching any movie. To help the user find the relevant information in a short time, a tool namely, RS has been developed by scientists/researchers (Jannach et al. 2010). The Netflix challenge encouraged researchers to comprehend the problem of recommendation as, given a 2-D matrix (users x items) where each cell represents the rating given by user to item, the task is to predict ratings of unrated items by users (Ricci et al. 2011; Xu et al. 2021;

✉ Niyati Baliyan
niyatibaliyan@igdtuw.ac.in

Saumya Bansal
saumyab271@gmail.com

[1] Indira Gandhi Delhi Technical University for Women, New Delhi, India

[1] https://repository.iimb.ac.in/handle/2074/19426.

Bedi et al. 2017). The performance evaluation of RS is done by computing Mean Absolute Error (MAE), i.e., the difference between actual and predicted ratings. This formulation is simple, easy to understand, and considers the user's task as static with time. However, with the ever-growing growth of OTT, RS should not only be developed with a single objective to minimize MAE but with multiple objectives, i.e., relevant short-term and long-term recommendations; diverse recommendations; fairness in terms of items' popularity; exploitation of sequential and non-sequential interaction of users with items; candidate generation and ranking.

Some recent recommendation approaches use sequential behavior of users to improve their consumption experience (Quadrana et al. 2018). The sequential approaches are designed to predict next items, i.e., short-term predictions at the expense of worse long-term predictions. The approaches using sequential behavior of users to predict next items are known as Sequential Recommender System (SRS). Given sequential behavior of users, the next item recommended by the RS in the sequence is a prediction that becomes the basis for further predictions and thus increases the chance of moving away from the user's taste as the sequence progresses. This results in sequential recommendations in terms of long and short-term predictions. Some recommendation approaches are oriented towards long-term while others at short-term predictions. Unfortunately, these approaches under-perform when it comes to developing RS from a multi-objective perspective.

Current SRS make use of user's consumption behavior and recommends n items to choose from. The user select an item from the list of recommendations to consume and then recommendations are re-generated based on user's history along with current taste. These recommendations are good for short-term i.e., for next item consumption but in long-term, user taste changes with time and hence previously generated $(n-1)$ recommendations are of no use. Therefore, there is a need to save the re-calculation time and recommend items keeping in mind short-term and long-term needs of the user. Moreover, current approaches does not take into consideration two important parameters i.e. diversity and popularity while generating list of recommendations.

To provide the best user experience, instead of providing candidate[2] generated by the model in an unordered way, an ordered list (ranking in descending order of relevance) of candidate should be recommended. Further, while building an ordered list, users should be presented with a diverse set of items to choose from, thus helping them to confront items that they may have not even thought of. Moreover, recommendations should not be inclined towards popular items leading to the popularity bias problem in RS. Beyond merely exploiting the sequential behavior to predict future trajectories, focus on short-term as well as long-term predictions along with other multi-objective perspectives should be diverted (Quadrana et al. 2018; Wang et al. 2019). To sum up, goal is *to design a recommendation algorithm that addresses sequential short and long-term predictions for diversified and popularity free recommendations*.

In this work, we exploit the sequential and non-sequential interaction of users with items that hold a lot of information about the users' evolving and vanishing interests. We worked on RS in two phases: Candidate Generation and Ranking. In the first phase, a hybrid kNN-GRU model exploits the sequential behavior of users to generate short-term predictions that aim to identify items, users may consume soon along with non-sequential behavior to generate accurate long-term predictions that identify items users will eventually consume. This approach inherently takes care of two major limitations of RS: diversity and popularity bias. In the second phase, a novel ranking method has been used to provide an ordered list of recommendations without giving undue advantage to the popularity and thus rating count of old items. The following are key contributions of the work:

1. As opposed to other sequential approaches, our model covers both phases of recommendations, i.e., Candidate Generation and Ranking showing an accuracy of 85%.
2. The usage of both sequential and non-sequential behavior provides relevant short-term as well as long-term predictions.
3. The data sparsity, diversity, and popularity bias in the recommended list is inherently handled by the model.

The paper is organized as follows: Sect. 2 gives details about the Related work. Section 3 explains the proposed approach. Section 4 discusses experiments and results. Conclusion and Future Work is discussed in Sect. 5.

2 Related work

Traditional matrix-completion setup in the field of RS has been given a push by the popular Netflix Context. It encouraged researchers to find predicted ratings of unrated items considering the static taste of the user, with the motive to minimize the difference between predicted and actual ratings (Bennett et al. 2007). Recently, researchers have started exploring sequential interaction of users and items, motivated by the need to study inherent patterns within the data (Gupta and Katarya 2021), explore RS beyond similarity computation or matrix completion, and need to find next relevant items to leverage the change in users' taste over time. Further, there is a need to maintain diversity within the recommended list and thus increase user satisfaction.

[2] The term "candidate" is used for items produced by the proposed approach before ranking, whereas the term "items" for all other items.

Table 1 State-of-the-art recommendation approaches

Work	Description	Key findings	Sequential approach (S)/ non-sequential approach (NS)/both
Gupta and Katarya (2019)	Collaborative Filtering (CF) technique has been largely worked upon by researchers	IBCF provided better results as compared to UBCF	S
	Two CF techniques i.e., User-based CF (UBCF) and Item-based CF (IBCF) were compared		
Wu et al. (2017)	Temporal information has been exploited about users and items using RNN, thereby predicting future behavioral trajectories	RNN showed improved prediction accuracy over previous methods and implicitly captured temporal patterns in rating data	S
Katarya and Arora (2020)	Enhanced Deep Neural Network by stacking Capsule Networks on Bi-RNN integrated with PMF (CapsMF)	Exploiting reviews along with ratings showed improved accuracy	NS
		CapsMF showed poor results on smaller datasets	
Hidasi et al. (2015)	Considering the disadvantage of short session-based recommendations, whole session has been modeled by introducing modifications to RNN	Significantly outperformed strong baselines such as item-kNN	S
Jannach et al. (2016)	Model finds predicted ratings for items using deep autoencoders	Highlighted dual motive of recommendations, i.e., value for user's experiences as well as profit for the company	NS
		Does not explicitly take into account temporal behavior but shows good accuracy	
Mohammadzadeh and Rathinasamy (2020), Mohammadzadeh and Kumbasar (2020), Mosavi et al. (2020)	The use of new fuzzy logic system has been made to improve stability and robustness	New fuzzy system can be explored in the field of RS	NS
Korean (2009)	Temporal dynamics are modeled to better know about users' changing taste	Relation among items is revealed by learning the influence of decay rate over time	NS
Wang et al. (2020a), Wang et al. (2020b)	Proposed a novel method namely, DivRec_LSH based on Locality-Sensitive Hashing technique and historical usage records	Highlight the need to pay attention to the diversity of the recommendation list along with the accuracy	NS
Ludewig et al. (2019)	A comparison among four neural and non-neural approaches with session on a variety of datasets has been laid	Comparison reveals limited benefits of applying neural approaches to session- based recommendations in terms of precision and recall as compared to simpler approaches like kNN	Both
Katarya and Saini (2021)	Repetitive consumption of similar items for long time without trying out something new can make the user feel bored	Depicted the importance of having diversity within recommendation list	NS
Li et al. (2017)	A neural attentive recommendation machine (NARM) is integrated with encoder-decoder architecture to provide session-based recommendations	Find the main purpose of the user's current session but do not pay attention to the immediate and long-term needs of users	S
Devooght and Bersini (2016, 2017)	Re-framed CF as a sequence prediction problem using LSTM for richer problem taking the evolution of users' taste into account	Depicted the efficiency of RNN in short-term predictions but suffers in the long-term run	S

Table 1 (continued)

Work	Description	Key findings	Sequential approach (S)/ non-sequential approach (NS)/both
Lathia et al. (2017)	From the viewpoint of diversity, three CF algorithms are evaluated	Highlighted the importance of temporal diversity and discouraged repeated top-N items recommended to use	S
Covington et al. (2016)	Explored Deep Neural Network architecture for recommending YouTube videos	Split the problem of recommendation into two phases: deep candidate generation and deep ranking model A trade-off between CPU time and hidden layers is analyzed to build an efficient model which outperformed classical machine learning models	NS
Smith and Linden (2017)	An overview of two decades of RS at Amazon is discussed	Usage and success of kNN resulted in 100-fold sales growth of the company Item-based kNN is algorithm of choice over user-based kNN	NS

We focused on several state-of-the-art recommendation approaches for a better understanding of key findings listed in Table 1.

To sum up, a recommendation model should serve the dual purpose of generating candidates as well as ranking them to generate an ordered list along with a multi-objective perspective of recommendations. Further, a model should be able to cater to the long-term as well as short-term needs of the user. The proposed approach takes care of this dual need with an aim to explore and quantify the effect of combining sequential and non-sequential approaches on long-term and short-term goals with implicitly handling diversity and popularity bias in the final recommendation list.

3 Proposed approach

This section discusses our proposed approach covering the recommendation model from a multi-objective perspective as shown in Fig. 1. We consider and work on both phases of recommendations, i.e., Candidate Generation and Ranking to provide the best user experience . The motive is to provide users with efficient recommendations, i.e., serving exactly what to consume next along with the long-term suggestion of what ultimately will be consumed keeping in mind that prediction accuracy alone does not reflect the quality of RS. The two phases of the proposed recommendation approach are as follows:

(A) Candidate generation

In our proposed approach, we use a fusion of RNN variants and kNN for modeling sequential as well as non-sequential interaction of users and items. Seeking to the advantage of RNN in learning sequential patterns and generate promising results in various domains ranging from machine translation, speech recognition, to text summarization, we investigated RNN in the field of RS to generate relevant next recommendations in short-term as well as long-term. RNN typically generates recommendations by using a softmax function where high probability denotes most relevant recommendations. Traditionally, RNN has been used to study sequential data and generate recommendations without marking a difference between short and long-term predictions. Rather than solving the recommendation problem to find ratings of unrated items by users, we intend to find items that users will consume in the near future and in extreme case what they will consume next. Further, exploring long-term predictions, we also find items that users will eventually consume. Unlike static environment, the distinction between long-term and short-term predictions is important, to take the advantage of designing motive of the sequence-based

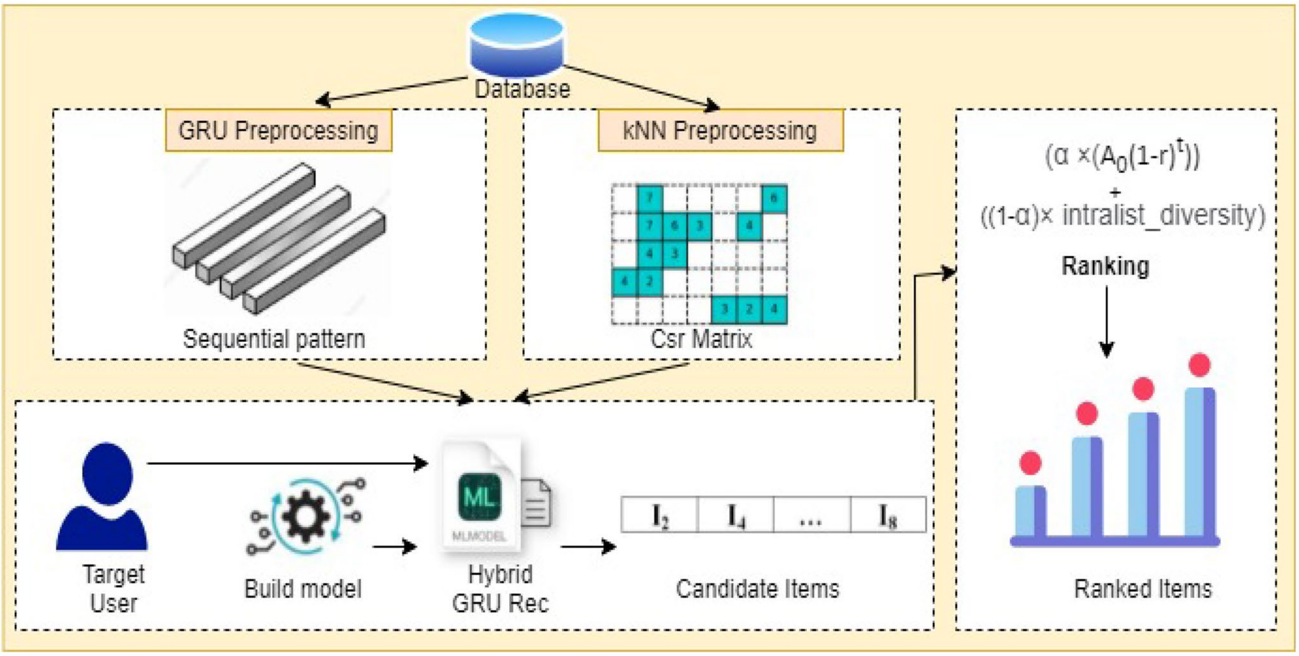

Fig. 1 Schematic illustration of hybrid GRU-Rec

model to find next items in the sequence, sometimes at the expense of long-term predictions which can be well-handled by the non-sequential model as discussed below:

1. Sequential Modeling

Similar to language modeling, to deal with sequential data, we consider the set of items as a vocabulary of words and watch behavior of each user as a sample sequence. Given a dataset of U users, we denote by S_t, sequence of rated items of user U at time t. The RNN learns from a sequence of items consumed by users with time as shown in Fig. 2. We have used variant of RNN i.e., Gate Recurrent Unit (GRU) in our approach with same basic illustration as shown in Fig. 2 but different cell. The GRU cell is shown in Fig. 3.

We develop a model that can make forward predictions, one item at a time. The next item to be consumed is recommended based on previous prediction in addition to the input sequence at each timestamp. For instance, at time $t1$, the input sequence of the user's interaction is split into two parts: the first part i.e., predictor consists of items consumed in sequence which is fed to the RNN, and second part is the label i.e., recommended item as shown in Fig. 4. At times $t2$, predictor will be I1, I2, I5, I4, I3, I12 in sequence and label will be the next predicted item, say, I9. This process is repeated until $'N'$ recommendations are generated.

During training, the model loops over these sequences to learn patterns hidden within the user's interaction. We exploit RNN's capability to automatically model the sequential information and make accurate forward predictions.

2. Non-sequential modeling

RNNs are capable of learning sequential data and predicting the next item to be consumed. However, these model fail at long-term predictions as next item's prediction is based on the previous sequence which includes predicted item as well. Resultantly, prediction accuracy reduces as sequence progresses. To tackle

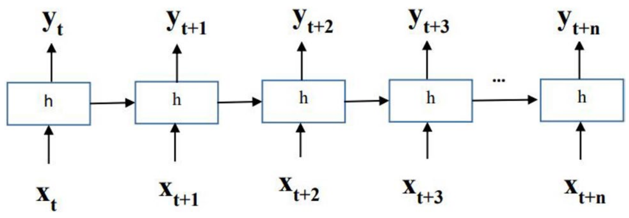

Fig. 2 Illustration of recurrent neural network

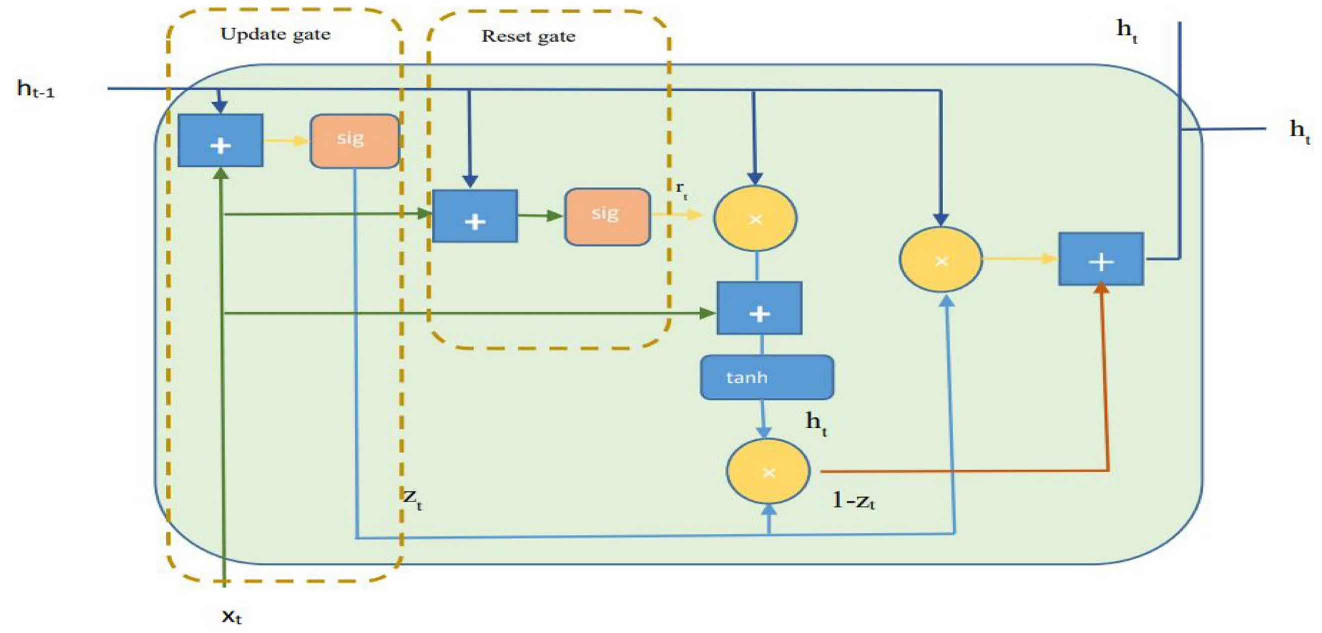

Fig. 3 Gated recurrent unit cell

this problem, kNN makes use of past user's behavior to generate recommendations. We chose item-based kNN over user-based kNN owing to its advantages (Smith and Linden 2017). For each unrated item of the target user, the model computes k nearest neighbors to compute its predicted rating. The items that have not been rated by the user are not included in the computation time and space.

(B) Ranking of candidates

Unlike traditional sequential RS, we take the output of both sequential and non-sequential model to generate a ranked list of N next items in the user's sequence. For each item $i\epsilon N$, a ranking score is calculated based on the item's similarity with other items in the list and its rating count with respect to decay rate. To balance intra-list diversity as well as popularity bias, we took a blend of both factors in the generation of the ordered list. The decay rate maintains the balance in rating count of the movie released in large gap, say, 1990 and 2000 as older the movie, higher the rating count. The final list of recommendations is generated based on the decreasing ranking score using equation (1).

$$\begin{aligned} rank(i) =& (\alpha \times (A_o(1-r)^t)) \\ &+ ((1-\alpha) \times intralist\_diversity) \end{aligned} \tag{1}$$

where,

$r =$ Decay rate in the range (0,1)

$\alpha = 0.5$ to mark balance between two

$t = 10$

$A_o =$ Total rating of an item

$intralist\_similarity =$ Diversity of item i with all other items in the list

The pseudo code is given below:

$$[I_1, I_2, I_5, I_4, I_3] \xrightarrow{\text{RNN Modeling}} [I_{12}]$$

$$\text{Predictor: } [I_1, I_2, I_5, I_4, I_3] \text{ Label: } [I_{12}]$$

Fig. 4 Illustration of splitting user's sequential interaction into two parts

Table 2 Accuracy analysis in terms of HR (%) and sps (%)

	ML 100K				ML 1M			
Models	HR@10	sps@10	HR@20	sps@20	HR@10	sps@10	HR@20	sps@20
kNN	0.61	0.11	0.60	0.22	0.57	0.28	0.53	0.37
LSTM-Rec	0.58	0.22	0.42	0.33	0.48	0.42	0.43	0.71
GRU-Rec	0.65	0.22	0.62	0.44	0.48	**0.71**	0.48	0.71
Hybrid LSTM-Rec	**0.66**	**0.33**	0.63	0.44	0.57	**0.71**	0.54	0.71
Hybrid GRU-Rec	**0.66**	**0.33**	**0.68**	**0.55**	**0.58**	**0.71**	**0.55**	**0.85**

Significant bold values are the most desirable values within a column, be it the minimum or the maximum value. In the case of a tie, multiple values are in bold

Algorithm 1: Hybrid GRU-Rec

```
Input: MovieLens dataset consisting of ratings provided by users to items in the range of 1-5
Output: List of recommended movies
$input\_sequence = []$
for i in corpus:
    $token\_sequence \leftarrow tokenizer.texts\_to\_sequences([i])[0]$
    for j in range(1, len(token_sequence)):
        n_seq ← $token\_sequence[: j+1]$
        input_sequence.append(n_seq)
    endfor
endfor
$max\_sequence\_len \leftarrow max([len(y)$ for y in input_sequence])
$input\_sequence \leftarrow np.array(pad\_sequence(input\_sequence, maxlen = max\_sequence\_len, padding =' pre'))$
$xs, labels \leftarrow input\_sequence[:, : -1], input\_sequence[:, -1]$
$ys \leftarrow to\_categorical(labels, num\_classes = total\_words)$
model ← $Sequential()$
model.add(Embedding(total_words, 64, input_length=max_seq_len-1))
model.add(GRU(512,dropout=0.05))
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
knn ← $NearestNeighbors(metric =' cosine', algorithm =' brute')$
for i in range(n_items):
    distances , indices ← $knn.kneighbors(csr\_sample[i])$
    indices ← $indices.flatten()$
    indices ← $indices[1 :]$
    similar_items.extend(indices)
    for j in similar_items:
        num += A[target_user-1][j]*(1-distances[0][k])
        k ← $k+1$
        den += (1-distances[0][k])
    endfor
    predicted_values[i] ← $column\_means[i] + (num/den)$
endfor
recommended_items ← $recommended\_items\_lstm[0 : n] + recommended\_items\_knn[0 : n]$
rank(recommended_items) ← $(\alpha \times (A_o(1-r)^t)) + ((1-\alpha) \times intralist\_diversity)$
```

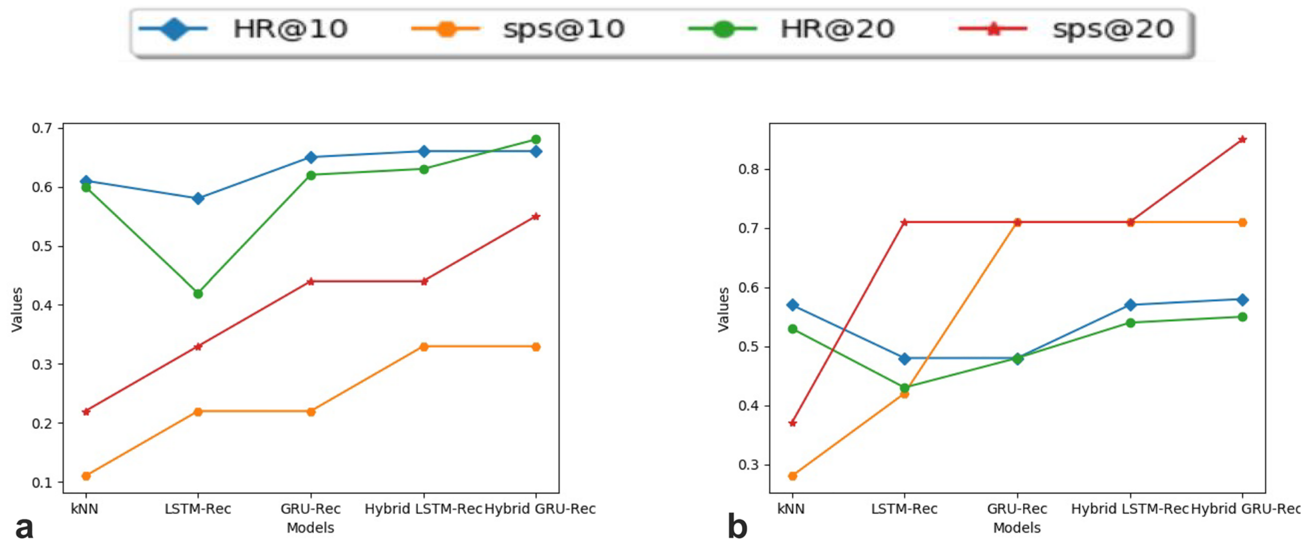

Fig. 5 **a** Comparison of four models on ML 100K. **b** Comparison of four models on ML 1M

Table 3 Analysis in terms of diversity (%) and blockbuster share (%)

	ML 100K				ML 1M			
Models	BS@10	Diversity@10	BS@20	Diversity@20	BS@10	Diversity@10	BS@20	Diversity@20
kNN	42.50	49.89	31.25	51.44	47.14	54.83	36.42	55.89
LSTM-Rec	16.26	62.55	**11.25**	**67.03**	**15.71**	**68.85**	12.87	70.45
GRU-Rec	**15.00**	**64.38**	13.51	65.10	**15.71**	**68.09**	**9.28**	**70.91**
Hybrid LSTM-Rec	30.00	56.99	29.37	58.27	34.28	57.94	34.28	62.38
Hybrid GRU-Rec	30.00	57.36	28.12	58.16	34.28	59.22	33.57	61.86

Significant bold values are the most desirable values within a column, be it the minimum or the maximum value. In the case of a tie, multiple values are in bold

The algorithm has been explained with the help of an example below:

Consider first 10 entries in the dataset are: (U2,I3,4), (U1,I4,5), (U4,I5,2), (U2,I4,4), (U1,I3,2), (U2,I6,4), (U1,I4,5), (U2,I9,2) where each entry is in the form of (User, Item, Rating). Firstly, these entries will be pre-processed to generate sequential consumption behavior for each user, for instance, for U2, sequential pattern will be [I3, I4, I6, I9]. Similarly, sequential consumption pattern will be formed for each user. These sequential patterns form the training set for GRU with the last item being marked as 'label' i.e., target. The same dataset is passed to kNN to generate recommendations for the target user. The recommendations generated from both GRU and kNN are merged and ranked based on novel ranking algorithm to provide relevant short-term (next item) as well as long-term (in sequence) recommendations.

4 Experiments and results

In this section, we present both qualitative and quantitative analysis of the proposed approach, Hybrid GRU-Rec with state-of-the-art and other hybrid permutation. We demonstrate the RNN models' ability to automatically model the next item in the sequence and kNN's capability to present relevant/ accurate items for long-term.

4.1 Datasets, parameters, and setup

In the following experiments, we use a single layer LSTM/ GRU with 512 cells along with one dense layer consisting of 100 cells. The *sigmoid* is used in the dense layer and *relu* as an activation function. For the non-sequential modeling, we used *cosine* similarity beacuse of data sparsity (Han et al. 2011) and chose 20 nearest neighbors. Detailed results can be found in Appendix.

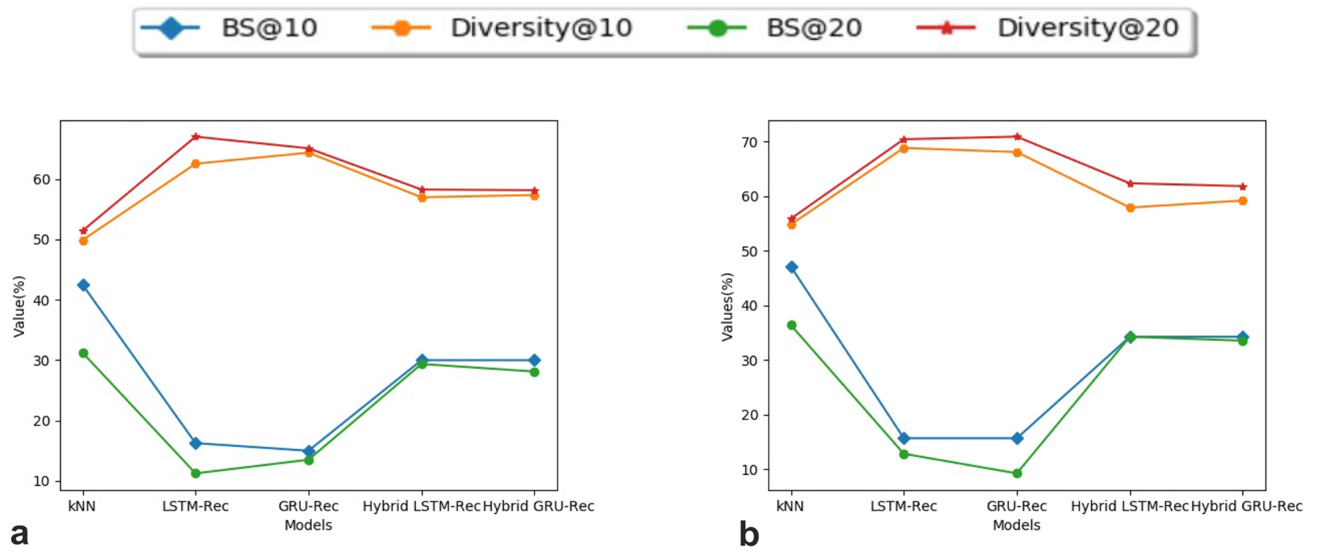

Fig. 6 **a** Comparison of four models on ML 100K. **b** Comparison of four models on ML 1M

Different public datasets are being used to aid researchers in their investigations on various algorithms. However, it was found that most research works have been centered about MovieLens dataset (GroupLens 2017) due to its ease of use and availability. Therefore, to study the effectiveness of the model built, we used the publicly available MovieLens datasets consisting of 100K ratings from 1000 users on 1700 movies and 1M ratings from 6000 users on 4000 movies. The datasets are 93.7% and 95.8% sparse respectively. Each user in the dataset has rated at least 20 items, therefore, the model based its recommendations on first 20 items consumed by the user and remaining items are used for evaluation purpose. Our model is evaluated on K20 GPU using 12GB RAM.

4.2 Competing methods

We compared our proposed approach against the following baseline[3] algorithms:

1. kNN-Rec (Smith and Linden 2017): It is one of the oldest collaborative filtering methods, which is static, yet produces excellent results in field of recommendations. It has been widely adopted by Amazon since 2003 and has shown great results in increasing its productivity. For comparative analysis, item-based kNN is considered over user-based owing to its advantages.

2. LSTM-Rec (Wu et al. 2017): It provides a compact model by exploiting sequential interaction among items and users using LSTM. The next k item recommendations are the ones with the highest transition probabilities from the last state.

3. GRU-Rec (Hidasi et al. 2015): It is a sequential model that utilizes whole session information to provide more accurate recommendations.

4.3 Evaluation metrics

To evaluate the RS quantitatively as well as qualitatively, we use several metrics (Devooght and Bersini 2017; He et al. 2018; Yuan et al. 2018) defined in this section.

1. Hit Rate (HR): It is the ratio of the total number of correctly recommended items to the number of users in the test data.

$$hitrate = \max_{i=1\ldots K} \begin{cases} 1, & \text{if } R_i \in T. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where R_i is the item in the list of recommendations and T is the set of items that are in the test set for the target user.

The average of this metric across users is typically called “Hit Rate”. In simplified form, hit rate can be defined as,

$$hitrate = \frac{\#hits}{U} \tag{3}$$

where U represents number of test users; #hits represents number of correctly recommended items

[3] All models have been implemented under the same runtime environment and renamed for the sake of comprehension.

2. Blockbuster Share (BS): It is used to evaluate the effect of popular items over recommendations. It finds the percentage of items recommended that are among the 1% of popular items in the dataset.

   $$\mathrm{BS} = |R \cap P_i|$$

   where P_i represents 1% of popular items based on rating count and R represents list of recommendations.
3. Short-term Prediction Success (sps): It measures the strength of RS in short-term predictions, i.e., exactly the next item. It returns 1 if the next item in the sequence belongs to the list of recommendations else 0.

   $$sps = \sum_{U} \begin{cases} 1, & \text{if next item } \in R_i. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

   The average of this metric across test users is taken.
4. Diversity: It is used to evaluate the variation in the recommendation list.

   $$diversity = 1 - similarity \tag{5}$$

   where similarity is defined as,

   $$similarity = \frac{\sum_{i=1}^{n} L1_i \times L2_i}{\sqrt{\sum_{i=1}^{n} L1_i^2}\sqrt{\sum_{i=1}^{n} L2_i^2}} \tag{6}$$

   where, L1 and L2 are recommendation sub-lists.

4.4 Results and analysis

This subsection discusses the performance of models in term of metrics defined in Sect. 4.3.

4.4.1 Accuracy analysis

A trade-off between long-term and short-term predictions is shown in Table 2 and Fig. 5 with HR representing long-term and sps representing short-term prediction outcomes. The HR and sps for top 10 and 20 items are depicted as HR@10, sps@10 and HR@20, sps@20 respectively. It is worth noting that the recommendation model can not be judged solely on the basis of hit rate. The sequential models provide impressive results in terms of sps. Further, it can be observed that HR@10 for sequential models is more as compared HR@20 depicting the decrease in correct predictions as sequence progresses. The Hybrid GRU-Rec model can predict the next movie watched by 55% users on ML 100 while 85% on ML 1M when 20 movies are recommended. Although, kNN can correctly predict a maximum of 61% of movies watched by the user but rest at 11% in predicting the next movie watched. Alternatively, maximum sps for kNN is shown on ML 1M, i.e., 37% with a hit rate of 53%.

It is also worth noting that kNN model does not fall behind LSTM-Rec and shows significant overall recommendation results but LSTM-Rec takes a lead in recommending the next movie watched which is of utmost importance from the user's perspective. Although kNN beats LSTM-Rec and GRU-Rec in hit rate but loses battle with Hybrid LSTM-Rec and Hybrid GRU-Rec in both metrics with Hybrid GRU-Rec performing the best.

Comparing sequential models i.e., LSTM-Rec and GRU-Rec, both perform well in terms of sps with a significant improvement seen in case of GRU-Rec marking sps@10 at 71% in contrast to 42% sps for LSTM-Rec at ML 1M. Further, GRU-Rec also outperforms LSTM-Rec in terms of hit ratio on both datasets. Taking advantage of both kNN and variants of RNN, Hybrid LSTM-Rec and Hybrid GRU-Rec outperforms both LSTM-Rec and GRU-Rec models. Further, Hybrid GRU-Rec shows superiority over Hybrid LSTM-Rec in terms of both sps and HR with highest being 85% on ML 1M and 68% at ML 100K datasets respectively.

Interestingly, from Fig. 5, sparsity might be linked to sps or in other words, predicting the correct next item to be consumed for more users. As the sparsity within the dataset increases (ML 1M is more sparse compared to ML 100K), sps increases. This may be due to the possibility of less options available after watching a movie, in case of sparse data, and therefore resulting in correct next prediction. More the option after watching a movie, more the ambiguity of next item suggestion, resulting in less sps.

To sum up, taking advantages of both kNN and GRU-Rec, Hybrid GRU-Rec outperforms all other models in addition to Hybrid LSTM-Rec.

4.4.2 Popularity and diversity analysis

To assess the presence of popularity bias in the recommendation list, we measured the availability of 1% popular items in the recommendations shown in Table 3 and Fig. 6. We found that kNN model tends to recommend more popular items than sequential models with a maximum BS of 47.14% on ML 1M. The lowest effect of popularity bias is observed in the GRU-Rec model on ML 1M with a BS of 9.28%. With respect to diversity, we observed that sequential models recommend less similar items with the highest diversity of 70.91% observed on ML 1M as compared to kNN with a diversity of 55.89% under the same conditions. It is not surprising that GRU-Rec and LSTM-Rec recommends more diverse and less popular items than Hybrid LSTM-Rec and Hybrid GRU-Rec, given a focus on the blend of kNN in hybrid approaches. Further, we also studied a trade-off between popularity and diversity. Interestingly, the model recommending less popular items, i.e., BS have high item coverage, i.e., diversity.

4.5 Discussion

From Tables 1 and 2, we found that some approaches might perform well in long-term prediction but performs poorly at short-term predictions and vice-versa. Models that provide the best HR and sps might not provide the best BS and diversity. Therefore, we conclude that for a real-world RS, it is not only important to evaluate a system from a perspective of correct items being recommended but from a multi-objective perspective that could provide recommendations that are relevant, diverse (least similar), and free from popularity bias at the same time. The existence of popular items is more in list recommended by hybrid approaches with recommending maximum of four popular items depicted in Table 2, as compared to stand-alone sequential approaches, which forms about 0.1% of items considered. In terms of run time complexity, kNN model does not include the training phase but takes 4mins and 8mins to generate recommendations for the target user on ML 100K and ML 1M dataset respectively. On the other hand, Hybrid LSTM-Rec takes 2.5 hours for training on ML 100K while 1.5 hours is taken by Hybrid GRU-Rec. The same pattern is observed on ML 1M dataset. Further, training time for both LSTM-Rec and GRU-Rec is approximately same as that of Hybrid LSTM-Rec and Hybrid GRU-Rec on both datasets. However, overall recommendation time is slighlty smaller in comparison of hybrid models due to their stand-alone architecture i.e., absence of any other model such as kNN. In addition, between both hybrid approaches, Hybrid GRU-Rec wins over Hybrid LSTM-Rec with slightly performing better in terms of all four metrics considered and significantly in training time taking an hour less. Making a global observation, our proposed Hybrid GRU-Rec performs the best out of models considered, dominating them in overall aspects.

5 Conclusion and future work

In this paper, we aim to build and explore a recommendation model from a multi-objective perspective. Considering a blend of sequential and non-sequential behavior of users, our model, named Hybrid GRU-Rec, improves over generating relevant short and long-term predictions, and implicitly handles the effect of popularity bias and diversity within the recommendation list. For a real-world model, our work shows the equal importance of diversity and popularity bias in addition to long-short term predictions in analyzing any recommendation model. Further, a novel ranking method has been used to provide an ordered ranked list of recommendations and thus save users from getting entangled in recommended items. Interestingly, we also explored the relationship between data sparsity and sps. The training time of deep learning sequential models is usually high and needs high configuration systems. In the future, we will try to work and optimize the same.

Appendix

See Tables 4, 5, 6.

Table 4 Analysis to set batch size and neurons

Model	Batch size	Number of neurons	Training accuracy (%)
GRU	128	128	38
		256	58
		512	87
	256	128	32
		256	66
		512	**94**
LSTM	128	128	41
		256	52
		512	81
	256	128	45
		256	58
		512	88

Significant bold values are the most desirable values within a column, be it the minimum or the maximum value. In the case of a tie, multiple values are in bold

Table 5 Results of different dropout setting batch size = 256 and neurons = 512

Model	Dropout	Training accuracy (%)
LSTM	0	82.84
	0.1	72
	0.2	61.58
	0.05	74.85
GRU	**0**	**85.82**
	0.1	75
	0.2	65.78
	0.05	79.25

Significant bold values are the most desirable values within a column, be it the minimum or the maximum value. In the case of a tie, multiple values are in bold

Table 6 Results of kNN on different neighborhood size

Number of Neighbors	MSE	RMSE
5	2.109	1.427
7	1.814	1.328
10	1.660	1.269
15	1.512	1.212
20	**1.447**	**1.187**
30	1.637	1.342

Significant bold values are the most desirable values within a column, be it the minimum or the maximum value. In the case of a tie, multiple values are in bold

Availability of data The datasets analysed during the current study are publicly available at https://grouplens.org/datasets/movielens/.

Declarations

Conflict of interest We have no conflicts of interest to disclose.

References

Bansal S, Baliyan N (2019) A study of recent recommender system techniques. IJKSS 10(13–41):10

Bedi P, Gautam A, Bansal S, Bhatia D (2017) Weighted bipartite graph model for recommender system using entropy based similarity measure. In: The international symposium on intelligent systems technologies and applications. Springer, Berlin, pp 163–173

Bennett J, Lanning S et al (2007) The netflix prize. In: Proceedings of KDD cup and workshop, vol 2007. New York, p 35

Covington P, Adams J, Sargin E (2016) Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems, pp 191–198

Devooght R, Bersini H (2016) Collaborative filtering with recurrent neural networks. arXiv preprint arXiv:1608.07400

Devooght R, Bersini H (2017) Long and short-term recommendations with recurrent neural networks. In: Proceedings of the 25th conference on user modeling, adaptation and personalization, pp 13–21

Grouplens (2017) Movielens. Retrieved from https://grouplens.org/datasets/movielens/

Gupta G, Katarya R (2019) Recommendation analysis on item-based and user-based collaborative filtering. In: 2019 International conference on smart systems and inventive technology (ICSSIT). IEEE, pp 1-4

Gupta G, Katarya R (2021) A study of deep reinforcement learning based recommender systems. In: 2021 2nd International conference on secure cyber computing and communications (ICSCCC). IEEE, pp 218–220

Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Oxford

He X, He Z, Du X, Chua TS (2018) Adversarial personalized ranking for recommendation. In: The 41st International ACM SIGIR conference on research & development in information retrieval, pp 355–364

Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2015) Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems: an introduction. Cambridge University Press, Cambridge

Jannach D, Resnick P, Tuzhilin A, Zanker M (2016) Recommender systems-beyond matrix completion. Commun ACM 59:94–102

Katarya R, Arora Y (2020) Capsmf: a novel product recommender system using deep learning based text analysis model. Multimedia Tools Appl 79(47):35927–35948

Katarya R, Saini R (2021) Enhancing the wine tasting experience using greedy clustering wine recommender system. Multimedia Tools Appl:1–34

Koren Y (2009) Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 447–456

Lathia N, Hailes S, Capra L, Amatriain X (2010) Temporal diversity in recommender systems. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval, pp 210–217

Li J, Ren P, Chen Z, Ren Z, Lian T, Ma J (2017) Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 1419–1428

Ludewig M, Mauro N, Latifi S, Jannach D (2019) Performance comparison of neural and non-neural approaches to session-based recommendation. In: Proceedings of the 13th ACM conference on recommender systems, pp 462–466

Mardukhi F et al (2021) Cfmt: a collaborative filtering approach based on the non negative matrix factorization technique and trust relationships. J Ambient Intell Hum Comput:1–17

Mohammadzadeh A, Kumbasar T (2020) A new fractional-order general type-2 fuzzy predictive control system and its application for glucose level regulation. Appl Soft Comput 91:106241

Mohammadzadeh A, Rathinasamy S (2020) Energy management in photovoltaic battery hybrid systems: a novel type-2 fuzzy control. Int J Hydrogen Energy 45(41):20970–20982

Mosavi A, Qasem SN, Shokri M, Band SS, Mohammadzadeh A (2020) Fractional-order fuzzy control approach for photovoltaic/battery systems under unknown dynamics, variable irradiation and temperature. Electronics 9(9):1455

Quadrana M, Cremonesi P, Jannach D (2018) Sequence-aware recommender systems. ACM Comput Surv (CSUR) 51:1–36

Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, pp 1–35

Smith B, Linden G (2017) Two decades of recommender systems at amazon. com. Ieee Internet Comput 21:12–18

Wang S, Hu L, Wang Y, Cao L, Sheng Q. Z, Orgun M (2019) Sequential recommender systems: challenges, progress and prospects. arXiv preprint arXiv:2001.04830

Wang L, Zhang X, Wang R, Yan C, Kou H, Qi L (2020) Diversified service recommendation with high accuracy and efficiency. Knowl Based Syst 204:106196

Wang L, Zhang X, Wang T, Wan S, Srivastava G, Pang S, Qi L (2020b) Diversified and scalable service recommendation with accuracy guarantee. IEEE Trans Comput Soc Syst

Wu C.-Y, Ahmed A, Beutel A, Smola A. J, Jing H (2017) Recurrent recommender networks. In: Proceedings of the tenth ACM international conference on web search and data mining, pp 495–503

Xu J, Tan L, Gong D, Liu F (2021). Extended matrix factorization with entity network construction for recommendation. J Ambient Intell Hum Comput:1–13

Yuan F, Karatzoglou A, Arapakis I, Jose J. M, He X (2019) A simple convolutional generative network for next item recommendation. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 582–590