Improving the Accuracy of Sequential Recommendation Using a Time-aware Architecture

Ruoxi Sun
Faculty of Engineering and Information Sciences
University of Wollongong
rs992@uowmail.edu.au

Jun Yan
Faculty of Engineering and Information Sciences
University of Wollongong
jyan@uow.edu.au

Fenghui Ren
Faculty of Engineering and Information Sciences
University of Wollongong
fren@uow.edu.au

Cong Cao
School of Management
Zhejiang University of Technology
congcao@zjut.edu.cn

Abstract

Sequential recommendation methods play a critical role in modern recommender systems because of their ability to capture the context of users' interactions based on their behaviours performed recently. Despite their success, we argue that these sequential recommendation approaches usually consider the relative order of items in a sequence and ignore important time interval information. The time interval can reflect timely changes in user interests by capturing more accurate trends in the evolution of the interests. The ignorance of time intervals will make it difficult for them to learn high-quality user representation. To tackle that, in this paper, we propose a time interval-sensitive mechanism for the sequential recommendation, and we incorporate this mechanism in GRU, named it 2Gated-TimeGRU. We utilize time intervals between two consecutive user interactions as an additional model feature, which enables more accurate modelling of user short-term preferences and temporal dynamics. Experiments on real-world datasets show that the proposed approach outperforms the state-of-the-art baseline models in capturing sequential user preferences and recommendation accuracy.

Keywords: Recommender System, Sequence Modelling, Time-Aware.

1. Introduction

The rapid expansion of online resources dramatically increases the need for effective recommender systems (RS) to filter information. A key factor for building effective and personalised RSs is to accurately understand users' current interests, which are intrinsically dynamic and evolving (Tang& Wang, 2018). For example, Figure 1 illustrates the comparative proportions of movie genre interactions by real-world users in two consecutive months. The chart shows that the proportion of 'Drama' genre movies interacted remains stable over these two months, indicating the user's long-term preference. However, other categories of interacted movies experience significant fluctuations. For instance, genres like 'Historical' and 'Suspense,' highly favoured in February, saw minimal viewership in March. Conversely, 'Comedy' movies, which had no interactions in February, exhibited a heightened interest in March. Therefore, when generating recommendations, it is essential to consider users' long-term and temporary (or short-term) preferences to provide recommendations that better align with their current needs. To achieve this goal, sequential recommender systems have been proposed to accommodate users' evolving interests and preferences better, aiming to predict the following items that users will likely interact with based on past interactions (Rendle et al., 2010; Chen et al., 2022). These systems consider the temporal order and dynamics of user interactions and utilize this sequential information to capture patterns in user behaviour. Various methods have been proposed to model such sequential dynamics in user interactions to make the sequential recommendation (Wang et al., 2015; He&McAuley, 2016). Recently, with the revival of deep learning techniques, many sequential recommendation models employing deep neural networks to handle the sequential problem have been proposed and achieved significantly improved recommendation performance (Donkers et al., 2017; Hidasi et al., 2016, 2018). These sequential models, such as Gated Recurrent Units (GRU), allow such models to capture and learn more complicated user interaction sequential patterns because of their extensive time series analysis capabilities (Beutel et al, 2018; Zhou et al, 2019). The basic paradigm of existing works on GRU-based sequential recommendation models is to encode a user's historical interactions into a vector, which represents the user's preference, and make recommendations based on this hidden representation (Chung et al., 2014). Among these sequential recommendation models, the GRU-based models are recognised as one of the most effective and popular methods in improving the performance of sequential recommendation.

One critical factor of the sequential recommendation is to generate higher-quality user representation which also refers to the sequential pattern of user interactions through their historical interactions (Xie et al, 2022). Generating recommendation items corresponding to the user's timely needs are accessible using these user representations. However, one crucial piece of information that has been overlooked by major existing works when developing user representation, the time interval between two successive interactions, which can effectively model the recency of user preferences and needs (Li et al., 2020). The time interval can reflect timely changes in user interests by capturing more accurate trends in the evolution of the interests. Traditional sequential recommendation methods only consider the relative order of items in sequence logs and ignore the time interval in the sequence. The ignorance of time intervals will lead to inaccurate recommendation results (Wang et al., 2021). For example, suppose two items have a shorter time interval between interactions. In that case, it may indicate that they are more relevant in the user's mind because the user interacts with them in quick succession, and vice versa. Figure 2 illustrates the movie interaction sequential log of the user depicted in Figure 1 over six days. Evidently, on the 14th and 15th of March, the user exhibited a stronger inclination towards watching Kung Fu and Drama genres. However, three days later, the user displayed a heightened interest in Comedy films in subsequent interaction records. Therefore, it is better to model the relevance between items by taking the time interval into account so as to improve the recommendations' accuracy.

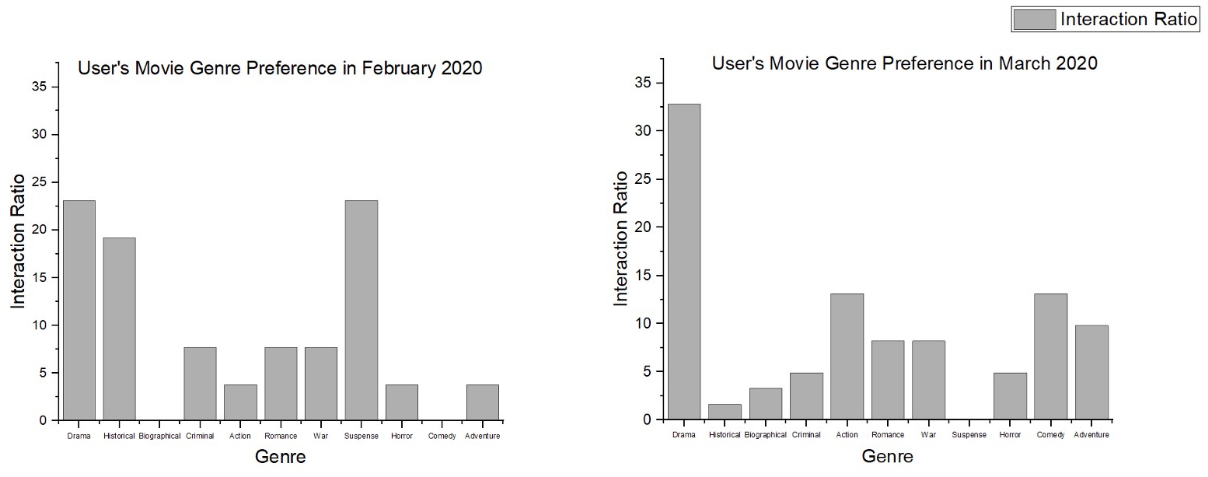

Fig. 1. Comparative Analysis of a User’s Movie Genre Preferences in February and March 2020

Due to the issues mentioned before, the time interval information in sequential recommendation task is less well studied. In this paper, we propose a time interval-sensitive mechanism for the sequential recommendation and then incorporate this mechanism into the GRU-based method to generate a more precise user representation. We name it as 2TimeGated-GRU. The critical mechanism in our time-sensitive GRU model corresponds to two time-aware gates. We utilize the gated tool in classical GRU to cope with the time-aware distance. Time gates can control to what extent the information should be transferred to future states according to time intervals. Two time gates are introduced in this time-sensitive sequential recommendation model, past time gate and future time gate. The past time gate captures the time interval between two consecutive historical interactions. In contrast, the future time gate captures the time interval between the current interaction and the interaction in the following (future) timestamp. This mechanism enables the recommendation model to be sensitive to the time intervals of user interactions. We utilize the output of the 2TimeGated-GRU to interact with the matrix formed by encoding candidate items to calculate recommendation scores for each candidate item, generating the recommendation list.

There are two contributions of this paper. First, we propose a mechanism that considers the time interval as an essential factor in modelling user preferences in the sequential recommendation to handle the time interval issue be-tween consecutive user interactions. Second, we incorporate this proposed mechanism in GRU and show how our proposed 2TimeGated-GRU approach model users' short-term preferences more accurately and obtain more accurate recommendations. We conducted extensive experiments on two different-scale public datasets, namely Movelens-1M and Movelens-20M. The experiment results demonstrate that our proposed model outperforms the baseline models.

The rest of the paper is organized as follows. Section 2 provides a comprehensive review and analysis of the existing literature in the field. We identify the gaps and limitations in previous studies, which our research aims to address. Subsequently, in Section 3, we present our proposed model, detailing its architecture. Following that, we describe the experimental setup, including the datasets, evaluation metrics, and methodology used to assess the performance of our model in Section 4. We then discuss the experimental results and provide insights into the findings. Finally, we conclude this work in Section 5.

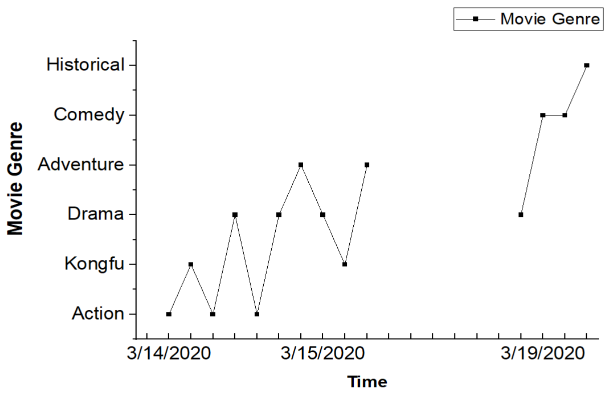

Fig. 2. User preferences at different time intervals.

2. Related Work

Early works on sequential recommendation usually model the sequential pattern with the Markov Chains (MCs) assumption (Shani et al, 2005, Rendle et al, 2010). Rendle et al. (2010) combine the power of the MCs and Matrix Factorisation (MF) to model both sequential patterns and general preferences to achieve a promising result in predicting the next interaction. In the subsequent work, to consider more previous interactions and extract more complicated patterns, the high-order MC is explored by He and McAuley, (2016). With the revival of deep learning, many studies have tried to adapt RNN and its variants, such as Long Short-Term Memory (LSTM) (Huang et al, 2019) and GRU (Hidasi et al., 2016) into sequential recommendations to explore the users' sequential patterns since they have shown to have an impressive capability in modelling sequences. The basic idea of these methods is to encode the user's previous records into a vector which refers to the user's sequential pattern (Wu et al, 2017). Compared with LSTM, GRU has fewer parameters, trains faster, and is less prone to overfitting (Yamak et al, 2019). GRU accomplishes this by integrating the forget and input gates of LSTM into a single update gate, which simplifies the architecture while still allowing it to capture long-term dependencies. For instance, Hidasi et al. (2016) adopt GRU modules to the session-based recommendation (a special case of sequential recommendation) task and named it GRU4REC. Later they improved GRU4REC with an improved sampling strategy and new loss functions in (Hidasi et al., 2018). There are also other following variants improving GRU4REC and achieving promising results by introducing attention mechanism (Kang& McAuley, 2018; Chen et al, 2019; Lv et al, 2019), hierarchical structure (Quadrana, et al, 2017) and user-based gated network (Donkers, et al, 2017). Furthermore, some follow-up works leverage GRU4REC to more precisely model users' short-term preferences by incorporating auxiliary information. These models integrate additional contextual information to achieve more accurate user representations, such as users' geographic locations (Liu et al, 2022) and knowledge graphs (Huang et al, 2018).

While the method of utilizing GRU to capture users' sequential behaviour patterns and generate recommendations has shown promising results, there is still a piece of important contextual information that is not effectively utilized in the current GRU-based sequential recommendation models when modeling user representations. The majority of them only consider the relative order of items in a sequence and ignore important time information such as the time interval and duration in the sequence, although interactions (or actions) close to one another in an interaction sequence have strong correlations. In particular, GRU-based sequential recommendation still limited representation power in time-vary user interaction. Traditional GRU methods treat each entry as having equal weight (Dey&Salem, 2017). However, as the user's subsequent interaction with the system may happen the next day, the next week, or the next month, considering consecutive user interactions with the system as an even distribution pattern, as done in traditional GRU-based RS, is unreasonable. The temporal distance deserves special handling. To address this problem, some recent work discussed this challenge. Wang et al. (2021) first leveraged time interval and duration information to capture users' preferences by adding an interval gate and a duration gate into the LSTM.

In another work, Lei et al. (2022) incorporated time interval awareness into GRU when modelling users' short-term preferences. The GRU model was made sensitive to temporal changes by adding a time interval-aware GRU model to extract short-term intention signals. However, these existing works only considered the time interval between two historical interactions in sequence logs, while the time interval between the current and next interaction is equally important. A shorter time interval between the present and next interaction may indicate that the user is still interested in a specific item or topic. In comparison, a longer time interval may suggest a decrease or shift in user interest. By comprehensively considering these time intervals, we can better understand user interests and behavioural patterns, enabling us to provide personalized recommendation services.

3. A Time-aware GRU-based Sequential Recommendation Model

The proposed time-aware sequential recommendation architecture utilizes time intervals between two consecutive interactions made by the user. Including time intervals in the GRU-based model as a feature to enable more accurate modelling of user short-term preferences and their temporal dynamics. Furthermore, by assigning different weights (time gates) to the input information based on the corresponding time intervals, the model can effectively adapt to the changing user behaviour patterns. As a preliminary, first, we will briefly introduce the formulas related to the traditional GRU.

According to the (Medsker&Jain, 2001) Standard RNNs update their hidden states using the following update function $h_t = g(Wx_t + Uh_{t-1})$. Where g is a smooth and bounded function such as a logistic sigmoid function, x_t is the unit's input at time t. An RNN outputs a probability distribution over the next element of the sequence, given its current state h_t. A GRU is a more elaborate RNN unit model that deals with the vanishing gradient problem (Cho et al., 2014). GRU gates essentially learn when and by how much to update the hidden state of the unit.

In the recommendation scenario, according to Hidasi et al. (2016), given the user u's recent interaction sequence $l = < (i^1, t^1), (i^2, t^2), \dots (i^T, t^T) >$, each element i^t in the list represent the interaction of user u with the item i at sequence index t, and t^t refers to the timestamp of the corresponding interaction. Through GRU, we can get the user u's sequence pattern at the timestamp t according to the following formula.

$$h_t^u = GRU(h_{t-1}^u, x_{i^t};\ \theta) \qquad (1)$$

In the above GRU formula, h_t^u is the GRU output, also refers to the sequential preference of user u at the timestamp t, and θ denotes all parameters of the GRU networks. x_{i^t} represents the embedding vector for item x_{i^t} where the dimension of items is the same as the embedding dimension of the hidden state h_t^u. The formulas of the GRU are as follows,

$$r_t = \sigma(x_t W_r + h_{t-1} U_r + b_r) \qquad (2)$$
$$z_t = \sigma(x_t W_z + h_{t-1} U_z + b_z) \qquad (3)$$
$$\tilde{h}_t = \emptyset(x_t W_h + r_t \cdot h_{t-1} U_h + b_h) \qquad (4)$$
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \qquad (5)$$

where $W, U \in R^{D \times D}$ are trainable parameters, D indicates the dimension of input embedding and hidden layers in GRU. r_t and z_t represent the reset and update gates, respectively, where the reset gate determines how to combine the current input with historical information, the update gate determines whether the input at the present moment will be updated to the new state. $\tilde{h}_t$ denotes the candidate hidden state, x_t represents the item input at time t, and $\cdot$ denotes the element-wise product. σ is the *sigmoid* function and ∅ refers to the *tanh* as the activation function. Through these gating mechanisms, GRU can effectively balance the trade-off between long-term memory and short-term memory, so as to better capture important patterns and information in sequence data.

As mentioned earlier, traditional GRU models treat each input uniformly. However, in RSs, user interactio ns occur at different times, and the timing of these inter actions plays a crucial role in understanding user prefer ences. For instance, when there is a long interval betwe en two consecutive interactions, the relevance of the pr evious preference information may diminish. e.g., let u ser u's interaction history be $l^u = \{(i^1, Feb\ 1^{st}), (i^2, Feb\ 2^{nd}), (i^3, Feb\ 12^{th}), \ldots\}$, it is more reaso nable to transmit more information from i^1 to i^2 than fr om i^2 to i^3 since i^2 is just just one day after i^1, while i^3 is 10 days after the last interaction. To tackle the probl em of user interaction time irregularity, we need to mo dify the gating logic in GRU to make it sensitive to tim e changes.

To make the traditional GRU sensitive to the interaction time interval, we introduce two time-aware features: past time feature δ_t and future time feature s_t in formulas (6) and (7). Then, the two time features are incorporated into the time gates in formulas (8) and (9), respectively, as follows,

$$\delta_t = \emptyset(W_\delta \log(t_t - t_{t-1} + \gamma) + b_\delta) \quad (6)$$
$$s_t = \emptyset(W_s \log(t_{t+1} - t_t + \varepsilon) + b_s) \quad (7)$$
$$t_\delta = \sigma(x_t W_\delta + \delta_t U_\delta + b_{t\delta}) \quad (8)$$
$$t_s = \sigma(x_t W_s + s_t U_s + b_{ts}) \quad (9)$$

Similar to the parameter structure of the reset and update gate, in the time gates, $W_\delta, W_s \in R^D$ and $W, U \in R^{D \times D}$. The future time feature s_t encodes the absolute temporal distance between the current state t_t and the prediction state t_{t+1}, while the past time feature δ_t encodes the relative temporal distance between two consecutive states that have already occurred. We add a fully connected layer to convert the time-aware features into dense vectors ($\delta_t U_\delta$ and $s_t U_s$), then compute time gates (t_δ and t_s) accordingly. Additionally, we have added two learning parameters, γ and ε, to the past time δ_t and future time feature s_t, respectively. This adaptively determines how much the new hidden state H_t matches the old state h_{t-1} versus how much it resembles the new candidate state $\widetilde{h}_t$. It is important to note that the two time-aware features defined in our approach are not fixed. The purpose of introducing these formulas is to incorporate the past time and future time information into the model's architecture. By incorporating these time-aware features, the model can capture the temporal dynamics of user behaviour and make more accurate recommendations. The presented formulas (6) – (9) are designed for the GRU-based model, and it is possible to adapt and modify these defined methods based on the specific requirements and characteristics of the recommendation model. Formula (5) is therefore updated to:

$$h_t = (1 - z_t \cdot t_\delta) \cdot h_{t-1} + z_t \cdot t_s \cdot \tilde{h}_t \qquad (10)$$

Through this, we can get the time-sensitive user u's sequential preference H_t^u at time t.

The core of RSs is the relevance-based ranking of items. Following Hidasi et al. (2016), our solution included pairwise ranking losses to obtain the recommendation list based on the user's sequential preference. The pairwise ranking compares the score, or the rank of pairs of a positive and a negative item and the loss enforces that the rank of the positive item should be lower than that of the negative one.

To generate the candidate next recommend items, we rank the candidate items by computing the recommendation ratings $R(u, i, t)$ using the formula (11)

$$R(u, i, t) = {H_t}^T \cdot X \qquad (11)$$

$R(u, i, t)$ is the rating function implemented as the inner product between the sequential preference representation matrix of users H_t at timestamp t and the embedding matrix X of the items in the system.

For parameter learning, we adopt the pairwise loss function following TOP1 in Hidasi et al. (2016), a regularized version of the Bayesian Personalized Ranking (BPR) (Rendle et al., 2012). Compared with the traditional BPR loss function, this TOP1 loss function add a regularization term $\sigma(\hat{r}_{s,j}^2)$, to punishes the score of negative samples. It alleviates the problem of synchronous growth of the scores of BPR positive and negative samples, so that the scores of positive and negative samples can become more stable,

$$L_s = \frac{1}{N_s} \cdot \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2) \qquad (12)$$

where N_s is the sample size, $\hat{r}_{s,i}$ $\hat{r}_{s,j}$ refers to the score on desired item i and the negative sample j.

4. Experiments

To objectively justify our theoretical models derived in the previous section, we conducted several experiments. The experiments aimed to answer the question of how our proposed 2Gated-TimeGRU performs on the sequential recommendation task when compared to a conventional GRU method, GRU4REC (Hidasi et al., 2016), as well as another time-aware state-of-the-art recommendation technique which considered only one-time gate (Lei et al., 2022), we named it Time-GRU in the following experiment section.

4.1 Methodology

In the following section, we will describe the evaluation metrics and datasets we used and the algorithmic setup.

Evaluation Metrics. Based on temporally ordered lists of consumed items, our objective is to correctly predict the next item a user will potentially interact with. A single user-item tuple, therefore, represents the ground truth at a particular time step. The target item should be among the first few recommended items to give the user adequate recommendations. From another perspective, the degree of discreteness in the predicted ranking matrix of the user-item pairs ($H_t{}^T \cdot X$) has also been verified for the three models. Following recent RS research, we use the following evaluation metrics:

Precision@k is a fraction of the top k recommended items relevant to the user. We set k = 1, 5, 10, 50 as it appears desirable from a user's perspective to expect the target among the first 50 items (Hidasi et al., 2016)). Moreover, it also can provide insights into the accuracy of the ranking to some extent. It assesses how well the recommended items are ranked among the top-k positions compared to the ground truth.

MAE (Mean Average Error) represents the mean of the absolute error between the predicted and observed values.

RMSE (Root Mean Squared Error) represents the sample standard deviation of the difference between the predicted and observed values (known as the residuals). The root mean square error is used to indicate the sample's degree of divergence.

Datasets. We ran our approach and all the baselines on the following two real-world datasets with different scales.

MovieLens 1M (Harper&Konstan, 2015): The MovieLens 1M dataset consists of 1, 000, 209 ratings assigned to 3, 900 movies by 6, 040 users. To mimic implicit data, in the data preprocessing, we binarized all ratings independent of their value, considering them as positive feedback (Rendle et al., 2012). Using the timestamps provided, we thus got an ordered sequence of consumption events for each user. We aimed to predict the next movie to watch.

MovieLens 20M (Harper&Konstan, 2015): The MovieLens 20M movie rating dataset consists of 20 million ratings applied to 27,000 movies by 138,000 users. Similar to MovieLens 1M, we utilize sequential logs as input to generate the candidate recommendation item in the next timestamp.

We split each dataset into three parts: First, the major fraction of every sequence serves as training data. Second, we use a validation set during training to measure performance on the hyperparameter set. These validation results determine, for instance, early stopping used in the learning phase to avoid overfitting (Hofmann, 2001). Finally, we use a different test set to measure actual performance after learning is completed. The training set consists of the first 80% of every user interaction sequence log. The remaining of each sequence is split evenly into validation and test sets while maintaining the order.

Hyperparameter setup. Table 1 shows the best-performing parametrizations we configured for the experiments. We optimized the hyperparameters by using the grid search method to run extensive experiments for each dataset and loss function. The best parametrization was further tuned by individually optimizing each parameter. The best-performing parameters were then used with hidden layers of different sizes. These parameters apply to the model proposed in this paper and two baseline models. The optimization was done on a separate validation set. Then the networks were retrained on the training plus the validation set and evaluated on the final test set.

Table 1. Best performing hyperparameter sets.

Dataset	Loss function	Batch size	Dropout	Learning rate	GRU unit
Movielens-1M	TOP1	100	0.1	0.1	512
	MAE&RMSE	100	0.0	0.05	32
Movielens-20M	TOP1	500	0.2	0.05	64
	MAE&RMSE	500	0.1	0.05	32

Table 2. MAE and RMSE results in MovieLens 1M and 20M datasets.

	Movielens-1M		**Movielens-20M**	
	MAE	RMSE	MAE	RMSE
GRU4REC (Hidasi et al., 2016)	0.8313	1.1112	0.7039	0.9222
TimeGRU (Lei et al., 2022)	0.8066	1.0687	0.6982	0.9148
2Gated-TimeGRU	0.7994*	1.0359*	0.6469*	0.8963*

4.2 Results and discussion

Table 2 shows the results (MAE and RMSE) received after training the baseline models as well as our proposed model on three datasets.

Figure 4 shows the comparison accuracy of our proposed model and the two baseline models of Top@K (K = 1, 5, 10, 50) prediction on the Movielens-1M and 20M datasets, respectively. The y-axis represents accuracy, and the x-axis represents the Top@K value. The comparative analysis of our proposed 2Gated-TimeGRU model, TimeGRU, and GRU4REC regarding Top@K accuracy reveals interesting insights. The results on the MovieLens-1M dataset clearly demonstrate the superiority of the 2Gated-TimeGRU model. Across different values of k, it consistently outperforms TimeGRU and GRU4REC. This improvement in precision is particularly pronounced in the top 5 and top 10 recommendations, where 2Gated-TimeGRU exhibits a significant advantage over the baseline models (at top@5, precision is 6% higher compared to TimeGRU and 10% higher compared to GRU4REC, while at top@10, it is 14% higher compared to TimeGRU and 18% higher compared to GRU4REC). This notable enhancement in precision underscores the effectiveness of the proposed model in capturing temporal dynamics and improving recommendation accuracy. However, on the MovieLens-20M dataset, the accuracy advantage of 2Gated-TimeGRU slightly diminishes compared to the smaller dataset (at top@5, precision is 3.8% higher compared to TimeGRU and 7% higher compared to GRU4REC, and at top@10, it is 3% higher compared to TimeGRU and 7% higher compared to GRU4REC). This can be attributed to the increased complexity and diversity of the larger dataset, which may introduce additional challenges in modelling user preferences accurately. Despite this, the line chart depicting the results still highlights the favourable performance of the 2Gated-TimeGRU model, as it consistently maintains higher accuracy than the baseline models across various values of k.

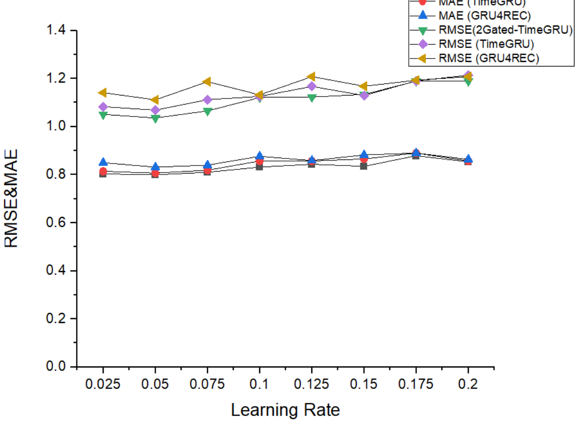

Figure 3. Comparative results on RMSE@MAE with different learning rates on the Movielens-1M datasets

Similarly, the same results are also reflected in terms of RMSE and MAE. From the contents of Table 2, it can be observed that among the three models, using 2Gated-TimeGRU to model users' sequential preferences consistently yields lower errors and more stable prediction results in the predicted ranking matrix. The experiment results reflect that the inclusion of two time gates allows for more precise modelling of the temporal dynamics of user behaviour, resulting in more accurate recommendations compared with the mechanisms that only considered one time gate and GRU4REC, which did not consider the time interval. As Koren (2020) has pointed out that even small changes in MAE and RMSE could lead to major differences in realistic recommendations. This demonstrates the effectiveness of incorporating multiple time intervals in the recommendation process. Moreover, compared with the traditional GRU-based sequential recommendation model, by modelling time intervals, the GRU model with two time-gating mechanism can adapt to changes in user preferences more effectively. It can capture drifts in users' interests, identify emerging trends, and avoid making recommendations based solely on historical data. This highlights the importance of considering time intervals for accurate and personalised recommendations. To investigate the impact of the learning rate as a hyperparameter on RMSE and MAE, we have included a plot that illustrates the variation in RMSE and MAE for the three models under different learning rates. It is evident that, across the majority of learning rates, our proposed model consistently demonstrates the highest level of accuracy compared to the baseline models, with its optimal performance occurring at a learning rate of 0.05. Furthermore, we conducted a time-based evaluation of the proposed and two baseline models. As shown in Table 3, although 2Gated-TimeGRU exhibits longer computation times compared to GRU4REC, this is primarily due to the additional time required for data retrieval, specifically, the retrieval of time stamps corresponding to interaction item IDs. In a horizontal comparison of the two time-aware models, 2Gated-TimeGRU demonstrates nearly identical computation times to TimeGRU in both Top@K and RMSE&MAE.

Table 3. Computational Time Comparison on Movielens-1M Dataset.

	Top@K	RMSE&MAE
GRU4REC	151S	922S
TimeGRU	318S	1403S
2Gated-TimeGRU	316S	1357S

5. Conclusion

In this research, we address the issue of traditional sequential recommender models, which only consider the relative order of items in user interaction sequences while ignoring crucial time interval information. In our study, we propose a time-aware sequential recommendation architecture that leverages the time intervals between two consecutive user interactions (both past and future intervals) to capture the time interval information. By assigning different weights to the input information based on the corresponding time intervals (time gates), the model effectively adapts to the evolving user behavior patterns. We then incorporate the proposed architecture into the popular sequential recommendation technique GRU, named 2Gated-TimeGRU. In the experiment section, we compared our proposed model with two baseline models on two publicly available datasets of different scales. The experimental results reflect the significance of modelling time intervals that can adapt to changes in user preferences more effectively. However, as previously introduced, the proposed model assumes that user preferences remain stable in shorter intervals. Therefore, this model may not be suitable for users with short-term preferences that tend to be more random.

One fundamental limitation is that our experiments were solely on movie domain data. While this allowed us to explore the effectiveness of our proposed methods within a specific con-text, it restricts the generalizability of our findings to other application domains. There-fore, future work should aim to conduct experiments on diverse datasets from various domains to validate the robustness and applicability of our approaches across different contexts. By expanding the scope of our experiments, we can gain a deeper understanding of the effectiveness and limitations of our proposed methods in real-world scenarios. Furthermore, the proposed model focuses solely on simulating user preferences at the item level without delving into user preferences for item features. Therefore, in future research, we will extend the model by investigating how to incorporate user preferences for item features at a more granular level. This will enable us to create more precise modelling of user profiles.

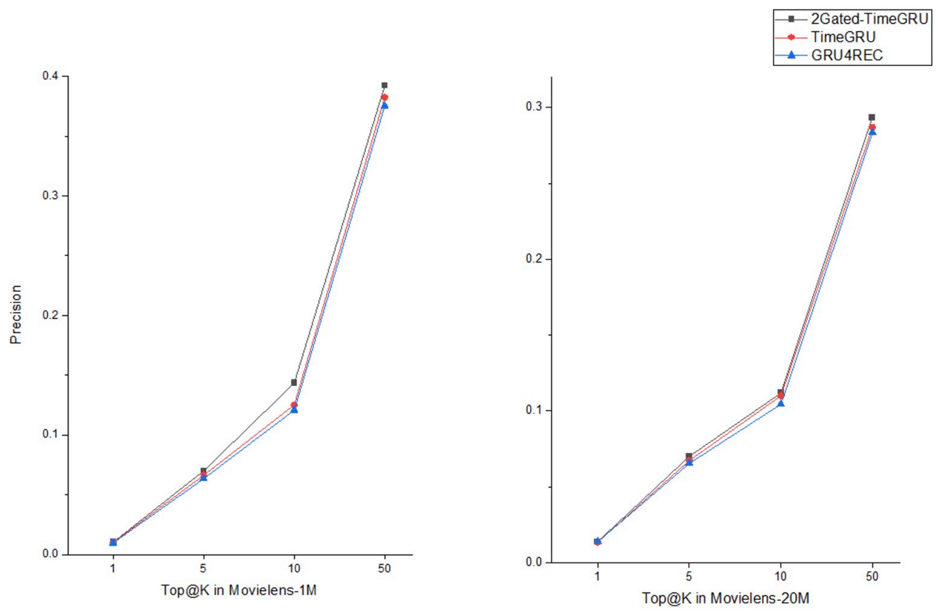

Figure 4. Comparative results on Top@K (K = 1, 5, 10, 50) prediction on the Movielens-1M and 20M datasets

6. References

Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V.,& Chi, E. H. (2018). Latent cross: Making use of context in recurrent recommender systems. *In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (pp. 46-54).

Chen, T., Yin, H., Chen, H., Yan, R., Nguyen, Q. V. H., & Li, X. (2019). Air: Attentional intention-aware recommender systems. *In Proceedings of 2019 IEEE 35th International Conference on Data Engineering (ICDE) Macao, Macao* (pp. 304-315). IEEE.

Chen, Y., Liu, Z., Li, J., McAuley, J., & Xiong, C. (2022). Intent contrastive learning for sequential recommendation. *In Proceedings of the ACM Web Conference 2022 Lyon, France* (pp. 2172-2182). ACM

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR, vol. abs/1409.1259* (pp. 1–9).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *In Proceedings of the Neural Information Processing Systems Workshop on Deep Learning. Montreal, Quebec*

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. *In Proceedings of the 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS) Boston, MA, USA* (pp. 1597-1600). IEEE.

Dien, T. T., Thanh-Hai, N., & Thai-Nghe, N. (2022). An approach for learning resource recommendation using deep matrix factorization. *Journal of Information and Telecommunication*, 6(4), 381-398.

Donkers, T., Loepp, B., & Ziegler, J. (2017). Sequential user-based recurrent neural network recommendations. *In Proceedings of the eleventh ACM conference on recommender systems, Como, Italy* (pp. 152-160). ACM.

He, R., & McAuley, J. (2016). Fusing similarity models with markov chains for sparse sequential recommendation. *In Proceedings of 2016 IEEE 16th international conference on data mining (ICDM) Barcelona, Spain* (pp. 191-200). IEEE.

Hidasi, B., & Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. *In Proceedings of the 27th ACM international conference on information and knowledge management, Torino Italy* (pp. 843-852). ACM

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. *In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico*.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42, 177-196.

Huang, J., Zhao, W. X., Dou, H., Wen, J. R., & Chang, E. Y. (2018). Improving sequential recommendation with knowledge-enhanced memory networks. *In Proceedings of the 41st international ACM SIGIR conference on research & development in information retrieval, Ann Arbor MI USA* (pp. 505-514).ACM.

Huang, L., Ma, Y., Wang, S., & Liu, Y. (2019). An attention-based spatiotemporal lstm network for next poi recommendation. *IEEE Transactions on Services Computing*, 14(6), 1585-1597.

Kang, W. C., & McAuley, J. (2018). Self-attentive sequential recommendation. *In Proceedings of the 2018 IEEE international conference on data mining (ICDM), Singapore* (pp. 197-206). IEEE.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD),* 4(1), 1-24.

Lei, J., Li, Y., Yang, S., Shi, W., & Wu, Y. (2022). Two-stage sequential recommendation for side information fusion and long-term and short-term preferences modeling. *Journal of Intelligent Information Systems*, 1-21.

Li, J., Wang, Y., & McAuley, J. (2020). Time interval aware self-attention for sequential recommendation. *In Proceedings of the 13th international conference on web search and data mining, Houston TX USA* (pp. 322-330).ACM.

Liu, X., Yang, Y., Xu, Y., Yang, F., Huang, Q., & Wang, H. (2022). Real-time POI recommendation via modeling long-and short-term user preferences. *Neurocomputing*, 467, 454-464.

Lv, F., Jin, T., Yu, C., Sun, F., Lin, Q., Yang, K., & Ng, W. (2019). SDM: Sequential deep matching model for online large-scale recommender system. *In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing China* (pp. 2635-2643). ACM.

Medsker, L. R., & Jain, L. C. (2001). Recurrent neural networks. *Design and Applications*, 5, 64-67.

Quadrana, M., Cremonesi, P., & Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4), 1-36.

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. *In proceedings of the Eleventh ACM Conference on Recommender SystemS, Como, Italy* (pp. 130-137). ACM.

Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. *In Proceedings of the 19th international conference on World wide web, Raleigh North Carolina USA* (pp. 811-820). ACM.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. *In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence. Montreal Quebec Canada* IEEE Computer Society (pp. 452–461). ACM.

Shani, G., Heckerman, D., Brafman, R. I., & Boutilier, C. (2005). An MDP-based recommender system. *Journal of Machine Learning Research*, 6(9).

Tang, J., & Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. *In Proceedings of the eleventh ACM international conference on web search and data mining, Marina Del Rey CA USA* (pp. 565-573). ACM.

Wang, D., Xu, D., Yu, D., & Xu, G. (2021). Time-aware sequence model for next-item recommendation. *Applied Intelligence*, 51, 906-920.

Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., & Cheng, X. (2015). Learning hierarchical representation model for nextbasket recommendation. *In Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval Santiago Chile* (pp. 403-412). ACM.

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, Cambridge United Kingdom* (pp. 495-503). ACM.

Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J. & Cui, B. (2022). Contrastive learning for sequential recommendation. . *In Proceedings of the 2022 IEEE 38th international conference on data engineering (ICDE), Kuala Lumpur, Malaysia* (pp. 1259-1273). IEEE.

Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019). A comparison between arima, lstm, and gru for time series forecasting. *In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence, Sanya China* (pp. 49-55). ACM.

Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C & Gai, K. (2019). Deep interest evolution network for click-through rate prediction. *In Proceedings of the AAAI conference on artificial intelligence, Honolulu, Hawaii* (Vol. 33, No. 01, pp. 5941-5948).