Multi-type Feature Mining and Fusion Model for Temporal Prediction

Wei Lu
School of Management and Economics
University of Electronic Science and Technology of China
Chengdu, China
luwei@uestc.edu.cn

Yan Hu
School of Management and Economics
University of Electronic Science and Technology of China
Chengdu, China
huyan@std.uestc.edu.cn

***Abstract*—Mining user behavior features is an essential part of recommender systems. As user preferences change dynamically, long-term interest features are very useful for predicting users' future behavior. Hence, the accuracy of temporal prediction will be greatly improved, if we can combine user behavior regularities with the item features. In this paper, we propose a long short-term memory (LSTM) variant, named Multi-type Feature Mining and Fusion (MFMF) model, to mine the long-term dynamic user behavior pattern and the potential relationships between items. We combine these features with user and item attributes to predict which items users will click next. Experiments on two real-world datasets, LastFM dataset and movie dataset, demonstrate the effectiveness of the proposed approach comparing to both traditional and state-of-the- art methods.**

***Index Terms*—Sequential data, LSTM, recurrent neural network, temporal prediction, behavior pattern mining**

I. Introduction

Personalized recommendation is critical to many web applications, such as music streaming, video watching, and e-commerce sites. The key to an effective recommender system is in modeling temporal preferences on items based on user historical activities [1], e.g. clicks. Hence, it is necessary and challenging to find the intrinsic patterns in the sequences of users' past behavior.

Previous studies have pointed out that both users' short-term and long-term interests are of great importance for recommendations [2]. Hence, designed architectures are required to distinguish and exploit these two types of interests simultaneously. For various online recommender platforms, there are three main difficulties to capture the relations of users' actions and predict their future behavior. First, user preferences are diverse and will change over time dynamically. The long-term features reflect users' general interest. It is important to exploit them, as it means that the recommended items should be influenced by users' past actions. Second, the user preference on items is not directly observable and has to be learned from implicit feedback. Recommendations based on traditional collaborative filtering of items often fail to achieve satisfactory results due to the large amount of recommended candidates on online platforms [3]. Third, local sequential interest should also be considered, which means that the recommended items should depend on most recently clicked ones. For example, if a user just clicks an episode of a popular drama, he is very likely to click that series in the near future, even if he has not shown interest in this type of video in the past.

Acknowledgement: This work is supported by National Natural Science Foundation of China (Grant No. 71901050) and the General Programs of the Sichuan Province Social Science Association (Grant No. SC19B035).

To solve the difficulties mentioned above, we consider fusing the long-term global representations of all users with the short-term activities of a particular user. Deep neural network is employed to mine global time-sensitive features in complex commercial scenarios [4]. We first learn the long-term evolution of users' preferences and find the potential connection between items according to the count of item usages. All items are mapped into a user interest space by using the historical click records. Since the number of predicted items is large, item features are very sparse. A general method is to embed the high-dimensional sparse space into a low-dimensional dense space. However, this method only encode items with one vector and can't mine the inner connections between items. Hence, skip-gram model is used for encoding.

Then, we learn the short-term preferences of a single user. We use the embedded expression of items in the user interest space as a feature of the item and combine it with other item attributes that will affect user behavior. Items that are clicked at the same time for multiple times are mapped to close positions. In order to capture the potential impact of different features automatically, we use the Factorization Machine (FM) framework [5] to capture cross features. In this way, both user's long-term interest characteristics and short-term preferences from the time-sensitive click sequences are learned simultaneously.

In summary, we make the following contributions:

- We propose a Multi-type Feature Mining and Fusion (MFMF) Model, which mines the time-sensitive product adoption data. By fusing multi-type features, the recurrent nature of usage dynamics is taken into consideration, which allows recommending personalized items to the right user at the right time.
- We show that the proposed model is applicable to handle the potential connection and similarity between items.

It can find users dynamic long-term interest and learn the regularities of their preference evolving based on historical click behavior.

- We conduct extensive experiments on two real-world datasets to demonstrate the effectiveness of our approaches.

The rest of this paper is organized as follows. Firstly, we give a review on feature mining and embedding. Then, the proposed framework, combining ID embedding, long short-term memory (LSTM) network and feature fusion is introduced in detail. Afterwards, the experimental results on two real-world datasets and conclusions are given.

II. The Proposed Method

The Multi-type Feature Mining and Fusion (MFMF) framework contains three parts: ID list embedding, long short-term interest memory networks, and multi-type feature fusion networks. The overall structure is shown in Figure 1. We will introduce each part in detail.

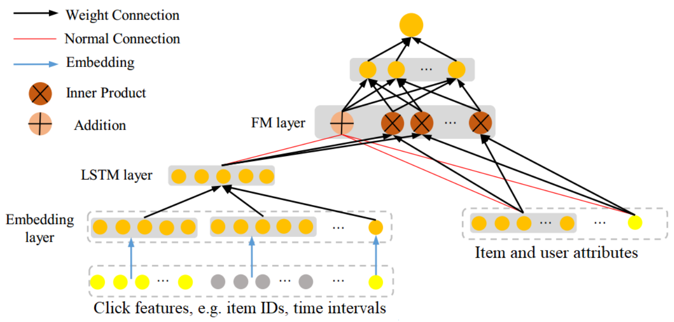


Fig. 1. MFMF network structure. At the embedding layer, items are mapped to vectors in the user interest space. The LSTM layer extracts the long-term behavior features of users. The FM layer part fuses user behavior features with short-term item features and feeds into fully connected layer for click prediction.

A. ID List Embedding

In order to capture the potential relationship and similarity between items, we consider adopting the skip-gram model of Word2Vec [6], commonly used in natural language processing. Assuming a user has clicked n items, the IDs of the n items form a sequence $(s_1, s_2, s_3, ..., s_n)$ in order of clicks. We put the historical click sequences of all users into the skip-gram model for pre-training, and get the item embedding values for further analysis.

Since words with similar semantics are closer to each other in an embedded space [6], we use the item embedding values to represent intrinsic relationships between items. The distributed semantic representation of words is learned from the click sequence. The goal of Skip-gram model is to learn item representations by maximizing the following probability distribution function P:

$$P = \prod_{s_t \in s} p(\mathcal{C}(s_t) \mid s_t) = \prod_{s_t \in s} \prod_{s_{t+j} \in \mathcal{C}(s_t)} p(s_{t+j} \mid s_t), \quad (1)$$

where s_t represents the ID of the current item and $\mathcal{C}(s_t)$ represents clicked items in the neighborhood of s_t. The function aims to let item s_t appear with context (neighbor) items in maximum probability. To calculate the maximum value of P conveniently, we change our objective to maximize the logarithm of P as follows:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} logp(s_{t+j} \mid s_t). \quad (2)$$

The hyper-parameter c is the length of the relevant forward and backward contexts of the click list, and the probability $p(s_{t+j} \mid s_t)$ is calculated as:

$$p(s_{t+j} \mid s_t) = \frac{exp(v_{s_t}^T v'_{s_{t+j}})}{\sum_{l=1}^{|V|} exp(v_s'^T v_{s_t})}, \quad (3)$$

where v_s and v'_s represent the input and output vector of the click list and v_{s_t} represents the input of s_t. V is the collection of all IDs in the dataset.

To improve training speed and the quality of the embedded item vector, we add some negative samples. The probability $p(s_{t+j} \mid s_t)$ can be expressed as:

$$p(s_{t+j} \mid s_t) = \prod_{z \in \{s_{t+j}\} \cup NEG(s_{t+j})} p(z \mid s_t), \quad (4)$$

where $NEG(s_{t+j})$ represents a negative sample subset generated when computing item s_{t+j}. The probability $p(z \mid s_t)$ is calculated as follows:

$$p(z \mid s_t) = \begin{cases} \sigma({v_{s_t}}^T \theta^z), & z \ is \ a \ positive \ sample; \\ 1 - \sigma({v_{s_t}}^T \theta^z), & z \ is \ a \ negative \ sample. \end{cases} \quad (5)$$

where θ^z is the frequency of item z in the sequence, and $\sigma(x) = \frac{1}{1+exp(-x)}$.

Hence, the final objective function is:

$$\arg \ max \ \mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0}^{z \in \{s_{t+j}\} \cup NEG(s_{t+j})} \log p(z \mid s_t). \quad (6)$$

After training the model, we can obtain a high-quality distributed vector to represent the IDs which achieves the distance of similar items that are very close in the low-dimensional dense space. That is, if two items are clicked by the same people consecutively many times, the embedding results will be very similar, indicating that the two items are similar to each other. So that we can find out the potential connection between items.

B. Long Short-Term Interest Memory Network

We use long short-term memory (LSTM) networks [7] to obtain the long-term behavior feature of users. We generate click sequences from each user's historical click behavior. These sequences contain the embedded item id of each click, as well as other attributes such as item type features and the time interval between two adjacent clicks. We set a threshold

N for the length of historical clicks sequence. It is assumed that the current click behavior is only related to the past N clicks. Thus, a larger value of N indicates a longer term impact on users' interests and hobbies.

We put the historical click sequences into the LSTM layer as it can learn long-term dependency information. The LSTM neural unit first determines which information should be discarded from the current state. Then the neural unit determines which new information should be stored into the current state. Finally, it updates the old state. The output h_t at time t is indicated as follow:

$$\begin{aligned} f_t &= \sigma(w_f \left[h_{t-1}, x^t_{emb}\right] + b_f), \\ i_t &= \sigma(w_i \left[h_{t-1}, x^t_{emb}\right] + b_i), \\ \widetilde{C_t} &= tanh(w_c \left[f_{t-1}, x^t_{emb}\right] + b_c), \\ c_t &= f_t * c_{t-1} + i_t * \widetilde{C_t}, \\ o_t &= \sigma(w_o \left[h_{t-1}, x^t_{emb}\right] + b_o), \\ h_t &= o_t * tanh(c_t), \end{aligned} \tag{7}$$

where h_{t-1} represents the historical information and c_{t-1} is the cell state at last time point. $\widetilde{x^t_{emb}}$ represents the embedded short-term clicked item. f_t,i_t,$\widetilde{C_t}$ represent the probability of forgetting, inputting, updating the state of hidden cells in the last layer respectively. $\sigma(\cdot)$ and $tanh(\cdot)$ are sigmoid and tanh activation function. c_t is the cell state at times tamp t and h_t is current output. The last output $h_T(T = N)$ represents long-term interest features.

C. Multi-type Feature Fusion Network

For multi-type feature fusion, we choose a factorization-machine (FM) [5] mechanism for obtaining cross features. The input features of the FM model are represented as $x =< x', h_T >$. It includes item and user attribute features x', and users long-term interest feature h_T. The output of FM layer y_{FM} is the summation of an additional unit and inner product units (Fig. 1), formulated as follows:

$$y_{FM} = w_0 + < w, x > + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} < V_i, V_j > x_{j_1} * x_{j_2}, \tag{8}$$

where V_i and V_j are parameter vectors. w is the parameter of x and d is the feature dimension. $w_0 \in R$ is a constant offset.

Then, we put all features into the fully connected layer and get the probability of the user click the given item. The output of the fully connected layer $y_{connect}$ and the final $\widehat{y}_{i,j}$ output is calculated as follow:

$$y_{connect} = W * y_{FM} + B, \tag{9}$$

$$\widehat{y}_{i,j} = \sigma(W' * y_{connect} + B'). \tag{10}$$

Sigmoid function is the activation function of the last layer. $\widehat{y}_{i,j}$ represents the probability of user i click item j in the next prediction time slot.

III. Experiments

In this part, we experiment on two real-world datasets, LastFM and Movie dataset. We use the proposed method MFMF to predict whether a user will click the given songs and movies at the next timestamp, and compare it with state-of-the-art methods. We show that MFMF can efficiently model the recurrent user-item interactions over time, and achieve higher performances than other temporal models.

A. Data Descriptions

1.LastFM Dataset: This is a music recommendation datasets for research, which is collected from Last.fm API [8]. This dataset includes two data tables. One is user historical click log, containing user, timestamp, artist and song tuples. The other is the user information table, which contains the gender, age and nationality information. The dataset includes 992 users and 108,000 songs. In our experiment, we select 10,000 songs randomly. There are 939 users who clicked on these songs with a total of 80,391 times. We take the users' last click as the item to predict whether the user clicks or not. All the click records in the data sets are positive samples. An equal amount of negative samples is generated randomly. In the experiment, the dataset is divided into two parts randomly, 85% of the samples are divided as training sets, and the remaining 15% are for test sets.

2. Movie Dataset[1]: This dataset contains the records of video click log collected from Tencent movie [9]. It includes 1489 users who clicked movies during the 20 days from July 23, 2014, to August 12, 2014. There are a total of 2193 movies and 19,287 click behavior records. Each click record includes time, user ID, movie ID, accumulated historical clicks, movie type, production country/region, director and starring role information. The data preprocessing of the movie dataset is the same as the LastFM dataset.

B. Comparison Models

We compare MFMF to the following methods: the classic model DNN [10], the popular model Wide & Deep Learning [11] and DeepFM [12], and the recently proposed DIN [13] and DIEN [14] model.

Since DNN, Wide & Deep Learning, and DeepFM are not designed for time series, we use the recently clicked items feature, the user feature, and the candidate item features as the input feature for item click prediction. The DIN and DIEN models are sequential recommendation models. We take the historical click sequences, user characteristics, and candidate item features as input features. The DeepFM model uses the FM framework to extract cross features, while other models do not.

C. Evaluations

We adopt accuracy (ACC), area under curve (AUC) and F1-score as evaluation metrics to evaluate the performance of different models. When we predict whether a user clicks the

[1]https://v.qq.com/movie/

item for the next time, ACC is used to evaluate the accuracy of models and F1-score is used to balance precision and recall values. In addition, our can also predict the probability of user next click item and we take the AUC as an indicator to evaluate the quality of the model. Each experiment is repeated 10 times and the average value as the final experimental results.

D. Results and Discussion

The results of the experiment on the two datasets are shown in Table 1. The item ID features input into LR, Wide & Deep Learning, and DeepFM models are preprocessed by skip-gram model.

Table I shows the results of LR, Wide & Deep Learning and DeepFM model are not very satisfying on the tested datasets. This could be due to the lack of capabilities for handling long-term interest features. The implementations of DIN and DIEN are based on the DeepCTR library. According to the description of the model in DeepCTR library[2], the item ID features input into DIN and DIEN models havent been processed by the skip-gram model. They are embedded according to the general feature processing method. DIEN can mine user long-term interest features. However, due to the item sparseness, the results of item embedding cannot well reflect the characteristics of the item. DIEN cannot perform well. The DIN model cannot learn the dynamic changes in users' preferences. It adopts the attention mechanism to learn the item that users are interested in according to user historical behavior data. As shown in Table 1, the accuracy and F1-score of the DIN model on both datasets are not high. Both DIN model and DIEN model cannot get the user click peculiarity.

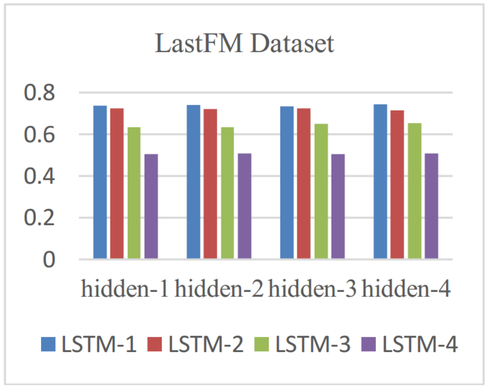
According to the experimental results on LastFM Dataset and Movie Dataset, the MFMF model proposed in this paper is superior to the previous models in terms of accuracy, F1-score and AUC. This indicates that MFMF is more precise in prediction and can recommend more correct items to the right user.

E. Hyper-Parameter Study

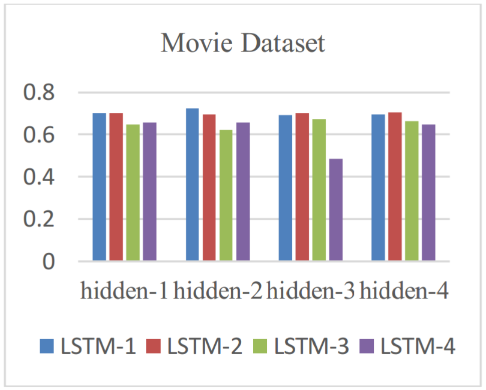
We performed hyper-parameter experiments on two datasets to evaluate the influence of different parts of MFMF by changing the value of one hyper-parameter while keeping the others fixed. The hyper-parameters in conducting experiments include: 1) number of LSTM and hidden layers; 2) number of neurons per layer; 3) dropout rate; 4) user history length.

1) Influence of LSTM and Hidden Layers: Hidden layers represent the fully connected layer between the output layer and the FM layer. Figure 2 and 3 shows the models accuracy when the number of hidden layers or LSTM layers changes on LastFM dataset and movie dataset. Hidden-1 denotes the number of the hidden layer is one. LSTM-2 indicates the number of LSTM layers is two. As shown in Figure 2, the number of hidden layers has little effect on accuracy. In Figure 3, when the number of hidden layers is small, increasing the number of hidden layers can improve the performance slightly.

[2] https://deepctr-doc.readthedocs.io/en/latest/index.html



(a) LastFM dataset



(b) Movie dataset.

Fig. 2. ACC comparison with different numbers of layers.

With the number of hidden layers increasing, the complexity of the model increases too so that the accuracy of the model decreases. From experiments on LastFM dataset, the accuracy decreases with the number of LSTM layers increasing, when the LSTM layer just has only one layer, the model has the highest accuracy. On Movie dataset, when the number of LSTM layer is one or two, the model has the best performance.

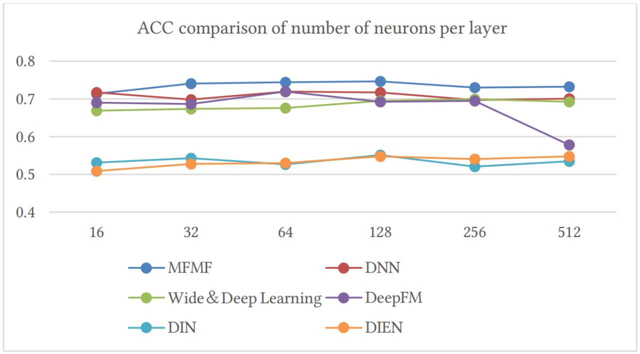
In summary, one LSTM layer or two fully connected layers are enough to achieve satisfactory accuracy.

2) Influence of Neurons per Layer: To learn more useful information, the accuracy of the model can be improved by increasing the number of neurons in each layer. However, too many neurons will also bring some problems. For example, the model will be more complex, which are prone to overfit. We exploit our model on LastFM dataset and compare the ACC and F1-score value of different models with different numbers of neurons in each layer. The number of neurons in each layer ranges from 2^4 to 2^9. Figure 3 shows the six different models' accuracy and F1-score values respectively.

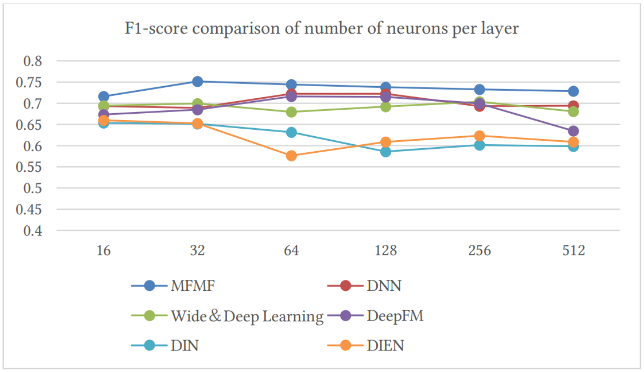
It can be inferred from the results that when the number of neurons is 2^4, the accuracy and F1-score of MFMF are lower. With the number of neurons increasing to 2^5, our model still has consistent performances. Hence, it suggests that with three layers of neural network, selecting more neurons will not make the model too complicated. In the above experiment, we

TABLE I
EVALUATIONS ON ACC, F1-SCORE AND AUC

	LastFM			Movie		
model	ACC	F1	AUC	ACC	F1	AUC
DNN	0.7189(± 0. 0161)	0.7228(± 0. 0155)	0.8947(± 0. 0160)	0.6807(± 0. 0128)	0.6837(± 0. 0081)	0.8612(± 0. 0145)
Wide & Deep	0.6997(± 0. 0180)	0.7042(± 0. 0107)	0.8892(± 0. 0024)	0.6614(± 0. 0101)	0.6599(± 0. 0204)	0.8429(± 0. 0043)
DeepFM	0.7199(± 0. 0231)	0.7160(± 0. 0147)	0.8861(± 0. 1283)	0.6889(± 0. 0115)	0.6789(± 0. 0138)	0.7995(± 0. 0641)
DIN	0. 5513(± 0. 0372)	0. 5863(± 0. 0287)	0.5538(± 0. 0522)	0.5211(± 0. 0108)	0.5765(± 0. 0610)	0.5473(± 0. 0544)
DIEN	0. 5478(± 0. 0233)	0.6089(± 0. 0233)	0.5703(± 0. 0491)	0.5447(± 0. 0327)	0.5948(± 0. 0620)	0.5196(± 0. 0210)
MFMF	0.7441(± 0. 0130)	0.7513(± 0. 0359)	0.9051(± 0. 0105)	0.7124(± 0.1200)	0.7346(± 0. 0207)	0.8777(± 0. 0100)

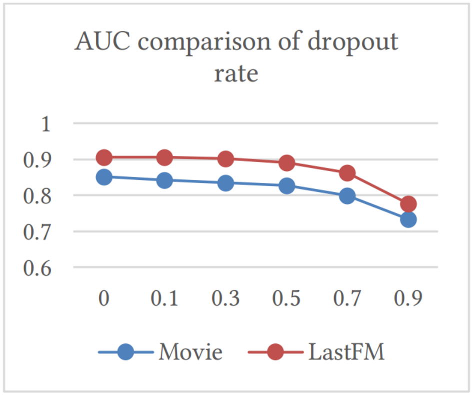


(a) ACC comparison

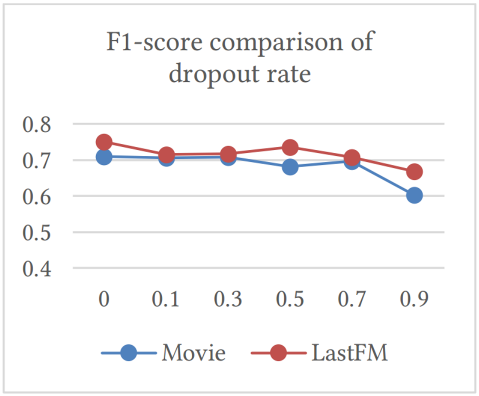


(b) F1-score comparison

Fig. 3. ACC and F1-score comparison of number of neurons on LastFM dataset.



(a) AUC comparison



(b) F1-score comparison

Fig. 4. AUC and F1-score comparison of the dropout rate.

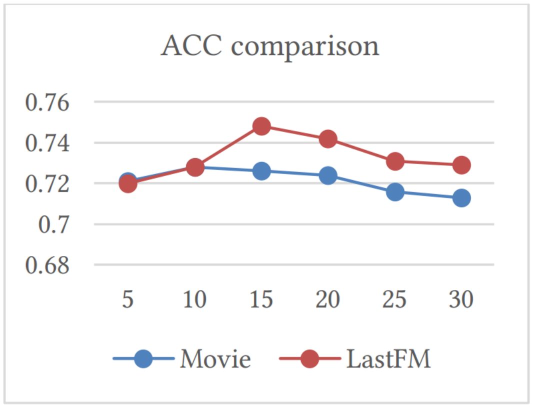
have found that three layers of neuron networks are enough to enable good performances.

Compared with other models, both ACC and F1-score value of MFMF model is higher. Only when the number of neurons per layer is 2^4, the accuracy of MFMF model is a little lower than DNN. This is because the number of neurons in MFMF model is too small to learn enough features.

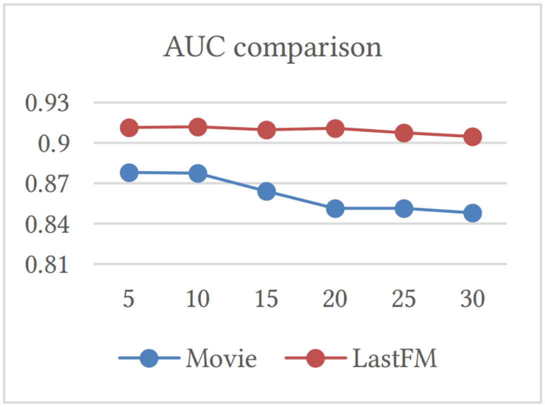
In summary, the MFMF model is robust and can reach higher accuracy comparing to other models.

3) Dropout rate.: Dropout technology [15] is used to delete connections of some neurons. It can reduce the complexity, but it may also reduce the accuracy because of the loss of some information. We set the dropout to be 0, 0.1, 0.3, 0.5, 0.7, and 0.9. The dropout rate value of 0.1 indicates that 10% of the links are deleted randomly. We use one LSTM layer, two hidden layers and 64 neurons for each layer. Experimental results are shown in Figure 4. With the increase of dropout rate, the AUC and F1-score values of the MFMF model change little until the dropout rate is up to 0.5. This shows the models robustness. When the dropout rate is more than 0.5, both the AUC and the F1-score decrease with the dropout rate increasing.

4) User History Length.: The user history length indicates how many records have been used for prediction. User clicks from a long period ago has few effects on current user click behavior because their interests have changed. In the Movie dataset, each individual clicks 12.95 movies on average. And only 40% of the people watched more than 5 movies. From the results shown in Figure 5, when the value of user history length is set to 5, some users long-term interest features have



(a) ACC comparison.



(b) AUC comparison.

Fig. 5. ACC and AUC comparison with different user history lengths.

not been learned. The accuracy of the model increases with the increasing of user history length. However, since most people watched only a small number of movies, the accuracy and AUC value of the model decreases with the increase of user history length after the value of user history length up to 10. In the LastFM dataset, people clicked 85.6 songs on average. About 73.89% of people clicked over 10 songs. The accuracy of the model first increases with the increase of user history length, and when user history length up to 15, the accuracy of the model decreases with the increase of user history length. The AUC values of the model almost have no change. Therefore, our model is relatively stable with enough historical user behavior data.

IV. Conclusion

This work introduces a novel recommender system that considers users' temporally dynamic characteristics globally. We develop an MFMF framework consisting of: 1) an embedding layer to learn the potential connection and between items; 2) an LSTM layer to mine long-term interest features; 3) a fusion layer to combine user-item attributes. Experiments performed on two real-world data demonstrate the effectiveness of MFM-F. The future work will focus on adding a more powerful high-order attribute interaction layer to strengthen the ability of feature fusion.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[2] D. Jannach, L. Lerche, and M. Jugovac, "Adaptation and evaluation of recommendations for short-term shopping goals," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 211–218.

[3] K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst, "Song recommendation with non-negative matrix factorization and graph total variation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2016, pp. 2439–2443.

[4] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 729–732.

[5] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011, pp. 635–644.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] M. Levy and K. Bosteels, "Music recommendation and the long tail," in *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*. Citeseer, 2010.

[9] W. Lu, F. Chung, W. Jiang, M. Ester, and W. Liu, "A deep bayesian tensor-based system for video recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, pp. 7:1–7:22, 2019. [Online]. Available: https://doi.org/10.1145/3233773

[10] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*. ACM, 2016, pp. 191–198.

[11] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 2016, pp. 7–10.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.

[13] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1059–1068.

[14] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5941–5948.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.