MARF: User-Item Mutual Aware Representation with Feedback

Qinqin Wang(✉), Khalil Muhammad, Diarmuid O' Reilly-Morgan, Barry Smyth, Elias Tragos, Aonghus Lawlor, Neil Hurley, and Ruihai Dong

Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland
{qinqin.wang,khalil.muhammad,diarmuid.oreillymorgan,barry.smyth, elias.tragos,aonghus.lawlor,neil.hurley,ruihai.dong}@insight-centre.org

Abstract. As deep learning (DL) technologies have developed rapidly, many new techniques have become available for recommender systems. Yet, there is very little research addressing how users' feedback for particular items (such as ratings) can affect recommendations. This feedback can assist in building more fine-grained user profiles, as not all raw clicks will truly reflect a user's preference. The challenge of encoding such records, which are typically prohibitively long, also prevents research from considering using the whole click history to learn representations. To address these challenges, we propose MARF, a novel model for click prediction. Specifically, we construct fine-grained user representations (by considering both the multiple items browsed, and user's feedback on them) and item representations (by considering browsing histories from multiple users, and their feedback). Moreover, the flexible up-down strategy is designed to avoid loading incomplete or overloaded historical information by selecting representative users/items based on their feedback records. A comprehensive evaluation on three large scale real-world benchmark datasets, showing that MARF significantly outperforms a variety of state-of-the-art solutions. Furthermore, MARF model is evaluated through an ablation study that validates the contribution of each component. As a final demonstration, we show how MARF can be used for cross-domain recommendation.

Keywords: Click-through rate prediction · Deep learning · Cross domain recommendation

1 Introduction

Predicting whether users will click on ads or items is a crucial problem in online advertising and recommender systems, where accurate predictions drive increased customer satisfaction and ultimately improve revenues [2,16,21,22]. Interestingly, the trend is that most recent solutions adopt deep learning techniques for click prediction.

Capturing user interests and constructing user profiles is an essential recommendation task. Normally, user interests are encoded into user embeddings,

T. Di Noia et al. (Eds.): ICWE 2022, LNCS 13362, pp. 3–15, 2022.
https://doi.org/10.1007/978-3-031-09917-5_1

which are randomly initialised at first and then optimised against records of user click behaviour. This conventional approach, while simple, does not improve recommendation quality as much as more recent sophisticated methods that either employ a sequence-based neural network with attention mechanism [5,21,22] to capture deep user interests through their behaviour history, or Graph Neural Networks [15,17] to generate richer user representations that capture both general and current interests.

Scope. We are interested in recommendation models to maximise CTR where the primary input is a dataset of implicit feedback from interactions [4] (e.g. dwelling time, number of views), while also leveraging available user and item metadata.

Problem Statement. We believe that relying strictly on user behaviour or click sequences from implicit feedback may not fully represent user interests. Besides, people are normally "cheated" to click an item by the attractive title/cover of the item and end up being dissatisfied [19]. In the movie domain, for instance, a user might watch several films of a particular genre but like a few of them; see Fig 1. Some of the clicking (e.g. 98 out of 337 drama movies) does not match users' favourites. So there is an opportunity to mine click histories for additional input signals that can complement the default implicit feedback used for click prediction. Such signals could be item ratings (explicit). For the sake of simplicity, we will henceforth refer to such complementary signals as **feedback**.

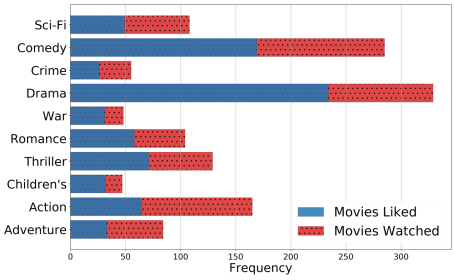

Fig. 1. Distribution of user preference cross different genres

In addition, much of the work in click-through rate (CTR) maximisation focuses on enriching user profiles to improve performance. So far, there is hardly any work that considers the whole click histories as a means of representing items. This is probably because it is more challenging to simultaneously encode long, dense histories for popular items and short click histories for fresh items in the long tail. This challenge also exists when building user profiles from click histories.

Therefore, to address these issues, we propose MARF, a new recommender for CTR maximisation that incorporates feedback when encoding user profiles. Through a novel flexible up-down sampling strategy, MARF is able to focus on representative interactions that produce richer representations to improve recommendation performance.

Contributions. The main contributions of this paper are:

- We introduce the feedback as an important feature that captures user and item properties from their interaction history.
- We propose MARF, a new recommender model for CTR maximisation that leverages feedback not only to produce richer user and item representations, but also to improve recommendation quality.
- Flexible up-down sampling strategy is proposed to choose representative users and items so that the computational costs and the impact of the long-tail problem are reduced.
- We conduct empirical tests to demonstrate the superiority of MARF over state-of-the-art models across multiple public datasets. We also present an ablation study to validate the utility of the various components in MARF.
- Finally, we show that MARF could potentially transfer knowledge across different domains with overlapping users or items.

2 The MARF Model

Our proposed, MARF model (depicted in Fig. 2) is split into three main components. Briefly, the modelling process starts from a set of inputs derived from user/item metadata and interactions. These inputs are used both to learn embeddings for users and items and as their feedback. Each user and item embedding

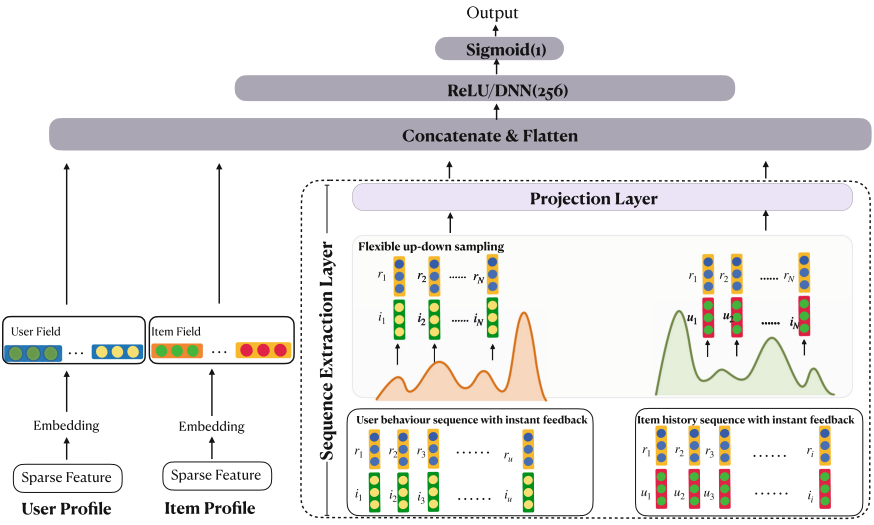

Fig. 2. The overall architecture of the proposed MARF model. The flexible up-down sampling strategy selects representative users/items with feedback to avoid loading incomplete/overloaded historical information. Projection Layer aims at projecting two different embedding into the same space and output the user/item representations

is concatenated with its matching feedback embedding. Separately, the *Sequence Extraction Layer* with our novel *flexible up-down sampling* strategy generates fine-grained representations from both user and item sequences. Finally, the outputs from the sequence extraction layer are merged with the embeddings learned from user/item side features and fed into a prediction output layer. In the rest of this section, we will describe these components in detail.

2.1 Model Inputs and Feature Representation

Overall, we use four forms of input for MARF, each composed from several sparse features. For any (u, i) interaction between user u and item i, we have these inputs: *UserProfile* provides a list of user attributes (e.g. gender, age, occupation) for the user u, whereas *ItemProfile* represents the features or metadata (e.g. color and category) of the item i. The *UserBehavior* is a sequence of (i_u, r_{ui}) tuples, where i_u is an item interacted with by the user u and r_{ui} is a feedback score (e.g. rating) assigned by the user u to item i. Similarly, *ItemHistory* is a sequence of (u_i, r_{iu}) pairs.

2.2 Embedding for User and Item Profiles

This component of MARF maps large sparse categorical features into low dense representations. In *UserProfile*, the k-th group of features such as occupation can be represented by $P_k \in \mathcal{R}^{V_k \times d_s}$, where V_k is the size of the sparse feature in *UserProfile* and d_s is the size of the sparse embedding. Similarly, in *ItemProfile*, the j-th group of features such as genres can be represented by $Q_j \in \mathcal{R}^{M_j \times d_s}$, where M_j is the size of the sparse features in *ItemProfile*.

At the same time, *UserBehavior* can be represented by $S_u = [i_1 : r_1; \ldots; i_{u_k} : r_{ui_k}; \ldots; i_{N_u} : r_{N_u}] \in \mathcal{R}^{N_u \times (d_i + d_r)}$, where N_u is the length of the user's behavior history, and i_{u_k} is the embedding of the item that the user interacts with at timestamp k, r_{ui_k} is the feedback embedding for item i from the user u, whereas d_i and d_r are the sizes of the item and the rating embedding respectively. Then we concatenate the embeddings of i_{u_k} and r_{ui_k} to construct user behavior at timestamp k. Similar to the *UserBehavior*, the *ItemHistory* is represented by $S_v = [u_1 : r_1; \ldots; u_k : r_{uk}i; \ldots; u_{N_v} : r_{N_v}] \in \mathcal{R}^{N_v \times (d_u + d_r)}$, where N_v is the length of the item's history, u_k is the embedding of the user, and $r_{uk}i$ is the feedback embedding for items i from the user u.

2.3 Flexible Up-Down Sampling

Most CTR models, including MARF, are characterised by a large number of parameters and are easily affected by the long-tail problem—80% of the data is comprised of information from only 20% of the users. For this reason, a large amount of the computational cost and the user and item historical sequence are significantly unbalanced. To tackle this problem, and improve memory consistency and computational efficiency, we propose the *flexible up-down sampling strategy*.

This flexible up-down sampling strategy is applied before each training step to reconstruct varying user and item historical sequences into a constant sequence. For instance, consider the example user profile in Fig. 1: a user is likely to have a propensity for certain kinds of items, thus we need not incorporate all items seen by the user into their behavior sequence. As such, for each user behavior sequence, we categorise items by their feedback (rating) and only sample a fraction of them per category as representative items. This operation is similar to the *stratified random sampling* [1] or *cluster-based sample* [20] that is commonly used to obtain representative samples from a set of entities. As a result, for any given entity (i.e. user or item) e, the size of their sequence—*UserBehavior* or *ItemHistory*—will be reconstructed to that of a constant sequence N. The number of samples we need to sample from each category is based on the portion of total number of interactions, such as $(N \times n_e^{(c)})/N_e$, where $n_e^{(c)}$ is the number of elements in category c and N_e is the total number of elements in the input sequence of the entity (i.e. user or item), whereas $N \in \mathbb{Z}^+$ is a hyper-parameter referring to the expected sequence length for the user or item that we need to reconstruct. It is important to note that the input sequence (user behaviour or item history) is concatenated with its respective feedback before the flexible up-down sampling strategy. Since different entities have different sequence lengths, the N hyper-parameter decides whether we perform an up-sampling or a down-sampling operation. If $N \geq N_e$, we up-sample, and if $N < N_u$ we down-sample.

Another noteworthy point is that not all users and items in datasets participate in the training process. This is because the flexible up-down sampler prioritises only representative users and items chosen from each category when reconstructing user and item sequence histories. For instance, Table 1 shows the percentage of users/items involved during the training process in three different datasets when the constant sequence length $N = 25$.

2.4 The Sequence Extraction Layer

To extract user interests, we begin from a set of item sequences with corresponding feedback, i.e. $S_u = [i_1 : r_1; \ldots; i_{u_k} : r_{ui_k}; \ldots; i_{N_u} : r_{N_u}]$. After flexible up-down sampling to get the reconstructed sequence indicated as S_{un}, we pass this input through an MLP fusion layer to project the item and the feedback embeddings into the same space to get $SE_u = [e_1; ...; e_i; ...; e_N] \in \mathcal{R}^{N \times d_e}$, where d_e is the fusion embedding size and N is the constant sequence length. Then we sum pooling the produced embeddings, SE_u, as the output for the user u's behavior representations SE'_u. We adopt a similar workflow to generate item history representations SE'_v.

2.5 The Prediction Layer

In previous sections we described how MARF learns embeddings from user and item features where a feature (e.g. genre) has multiple values (e.g. crime, fiction), the embedding process (described in Sect. 2.2) learns separate embeddings

for each feature value, and then the feature's embeddings are computed as the average of all the embeddings of its feature values. These transformed *profile embeddings* along with those learned by the Sequence Extraction Layer are then concatenated and fed into an MLP, with a final sigmoid function to predict the probability of the user liking an item.

We adopt the most widely used loss function in CTR prediction, the negative log-likelihood function defined as:

$$L = -\frac{1}{N} \sum_{(x,y)\in\mathcal{D}} (y \log p(x) + (1-y) \log(1-p(x))) \tag{1}$$

where $\mathcal{D}$ is the training set of size N, with x as input of the network, $y \in \{0, 1\}$ represents if the user liked the item and $p(.)$ is the final output of the network representing the prediction probability that the user likes the item.

3 Experiments

In this section, we compare the performance of MARF against that of several *state-of-the-art* models on three public datasets. We conduct an ablation study to verify the efficacy of each MARF model component.

3.1 Datasets

For the purpose of availability, we select datasets containing explicit ratings as an feedback feature. Table 2 summarises the key statistics of the datasets.

The **Amazon dataset** [14] contains ratings, product reviews and metadata from Amazon, and is used as a benchmark dataset in [9]. We use a subset named *musical instrument* which contains 903,330 users and 112,222 items, 1,512,530 samples and 505 categories. Due to sparsity, we adopt the k-core pruning method [6] to filter short profiles and only keep users with at least 20 ratings. We include *item style*, *category*, and *price* as features during training.

We selected **ML1M** and **ML20M** due to their familiarity to recommender systems researchers. **ML1M** contains 6,040 unique users, 3,706 unique items and 1,000,209 samples. We use *genre*, *zipcode*, *gender*, *age*, and *occupation* as side

Table 1. Percentage of user/ item involved in training

Datasets	Involved users(%)	Involved items(%)
Amazon Music Instr.	98.39%	79.58%
ML1M	89.64%	89.43%
ML20M	38.16%	58.16%

Table 2. Statistics of datasets

Dataset	Users	Items	Features	Samples
Amazon Music Instr.	903,330	112,222	510	1,512,530
ML1M	6,040	3,706	26	1,000,209
ML20M	138,493	26,744	23	20,000,263

features. **ML20M** is composed of 138,493 users, 26,744 items and 20,000,263 samples. The *genre* attribute is used as a side feature.

The statistics of the above datasets are summarized in Table 2. For all datasets, we train test split based on [12,16] where we randomly select 80% of samples for training and split the rest into validation and test datasets with equal size. We use the validation dataset for hyper parameter tuning. Each experiment is repeated 5 times, and the average performance with standard deviation is reported on the hold out test dataset. For all datasets, we treat samples with a rating less than 3 as negative samples, taking the lower score to indicate user dislike. Similarly, we treat ratings greater than 3 as positive samples. Samples with a rating of 3 are treated as neutral and removed from all datasets.

3.2 Baselines

In this section, we introduce the *state-of-the-art* baseline models chosen for comparison with MARF:

- **CCPM** [13] uses convolutional layers to capture partial dependencies between input features. It also turns the pooling layer into flexible p-max pooling to deal with flexible length of input.
- **NFM** [10] uses a second-order interaction layer called bi-interaction and a sum pooling layer to capture high-order feature interactions.
- **Wide&Deep** [2] is popular in production, and uses a wide network for cross product features while learning feature dependencies in its deep network.
- **DeepFM** [8] is an enhanced version of Wide&Deep where the wide part is replaced by a factorization machine.
- **AutoInt** [16] employs a self-attention mechanism to learn higher-order feature interactions.
- **FiBiNet** [12] learns feature importances using a Squeeze-Excitation Network (SENET), and feature interactions using inner product and hadamard product.
- **DIN** [22] uses local activation units to learn user interest representations from click histories.
- **DIEN** [21] employs an interest extractor (GRU) layer to capture users' temporal interests and an interest evolving layer (attention mechanism) to capture the change in interest that is relative to the target item.
- **AFN** [3] propose a new framework to learn arbitrary-order cross features adaptively from data so as to learn useful cross features from data adaptively, and the maximum order can be delivered on the fly.

3.3 Evaluation Metrics

We use two metrics in our evaluation: *AUC* and *Log Loss*.

- **AUC**: is a widely accepted metric for CTR tasks. It measures the probability that a random positive sample is ranked ahead of a random negative one [7]. A higher score denotes better performance.

– **Log loss**: is widely used in machine learning for binary classification tasks. It measures the difference between two distributions. The lower bound of log loss is 0, which indicates that there is no difference between two distributions. The lower value indicates better performance.

It is noteworthy that, in CTR prediction tasks, a slightly higher AUC or a lower log loss results in a significant boost for production systems [8,18].

3.4 Hyperparameters

In all embedding layers, regardless of the evaluation dataset, the dimension of the *feedback*, *user ID* and *item ID* is fixed at 200. We apply a one layer MLP for both user and item sequence extraction layer where the size is 256. The dimension of other sparse features is 56. For ML1M datasets, the model converges around 100 epochs, while the other datasets are run for 130 epochs.

Hyperparameter Search. We conducted a grid search on the ML1M dataset to find the constant sequence length value N. Figure 3 shows the AUC score and log loss on the validation split. Clearly, for ML1M, $N = 25$ has the highest AUC score, and its log loss is within bounds of the lowest log loss observed during the grid search. We keep the same constant length value for other the two datasets. For the optimization method, we use Adam with a mini-batch size of 1024 for both ML1M and ml20m, and 256 for amazon musical instrument dataset. The learning rate is set to 0.0001. The DNN layers are set to 2 with the size of the middle layer set as 256. The hidden layer activation function is ReLU and sigmoid is used for the output. For all baseline models, we apply Adam learning algorithm with the learning rate $\lambda = 0.001$. In the output layer of all baselines models, we apply two layers of DNN hidden units with sizes of 256 and 128 respectively. With the AutoInt, we apply a 3 layer attention structure with two heads for training to achieve the best results. We fine-tuning all baseline models with sparse feature dimension at parameters [4, 6, 8, 10, 15] and report the best results in validation set with dimension setting (after each result of the

Table 3. Performance comparison between MARF and eight baselines, showing MARF's superiority across three datasets.

	ML1M		Amazon review		ML20 M	
Model	AUC	Log loss	AUC	Log loss	AUC	Log loss
CCPM	0.8657 ± 0.002 (15)	0.4058 ± 0.0058	0.8029 ± 0.0117 (15)	0.2882 ± 0.037	0.8825 ± 0.0008 (10)	0.3491 ± 0.001
NFM	0.8843 ± 0.0008 (4)	0.3436 ± 0.0034	0.8239 ± 0.0115 (4)	0.2717 ± 0.0134	0.886 ± 0.0011 (4)	0.3481 ± 0.0038
WideDeep	0.8864 ± 0.0007 (4)	0.3339 ± 0.0023	0.8424 ± 0.0086 (8)	0.234 ± 0.0159	0.8875 ± 0.0003 (8)	0.3395 ± 0.0011
DeepFM	0.8854 ± 0.0012 (4)	0.3343 ± 0.0026	0.8297 ± 0.0083 (4)	0.312 ± 0.0343	0.8878 ± 0.0006 (4)	0.3426 ± 0.0009
DIN	0.8625 ± 0.0011 (8)	0.3343 ± 0.0017	0.8219 ± 0.0093 (8)	0.2650 ± 0.0137	0.8762 ± 0.0013 (8)	0.3432 ± 0.0003
DIEN	0.8723 ± 0.0039 (8)	0.3371 ± 0.0003	0.8135 ± 0.0086 (8)	0.3019 ± 0.0235	0.8804 ± 0.0002 (8)	0.3522 ± 0.0028
AutoInt	0.8863 ± 0.001 (4)	0.3355 ± 0.0022	0.8279 ± 0.0064 (10)	0.3051 ± 0.0327	0.8867 ± 0.0004 (10)	0.3444 ± 0.002
FiBiNet	0.8821 ± 0.0009 (10)	0.3852 ± 0.0033	0.8244 ± 0.0099 (10)	0.3486 ± 0.0205	0.8855 ± 0.0005 (10)	0.357 ± 0.0034
AFN	0.8884 ± 0.0008 (10)	0.3264 ± 0.0015	0.8206 ± 0.0049 (10)	0.1972 ± 0.0072	0.8871 ± 0.0003 (15)	0.3307 ± 0.0007
MARF (our)	$\mathbf{0.8968 \pm 0.0007}$	$\mathbf{0.3193 \pm 0.0005}$	$\mathbf{0.8462 \pm 0.0137}$	$\mathbf{0.1682 \pm 0.0113}$	$\mathbf{0.8958 \pm 0.0021}$	$\mathbf{0.3242 \pm 0.0003}$

AUC score), are shown in Table 3. The code used for this work is available on github.com[1]

3.5 Performance Comparison

In this section, we summarize the hold-out test performance of the selected algorithms on the ML1M, ML20M and Amazon datasets. For all baselines, we use the validation dataset for hyper parameter tuning, and report results on the hold out test dataset. From the results, shown in Table 3, it is clear that MARF significantly outperforms the baseline models on both the ML1M and Amazon datasets. For the ML20M dataset, all baseline models achieve similar performance, but MARF outperforms them marginally.

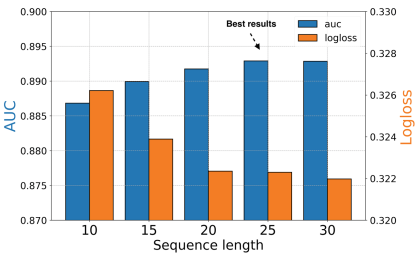

Fig. 3. Grid search sequence length for sampling

3.6 Ablation Study

Despite demonstrating strong empirical results, so far we have not isolated the specific contribution of each component of MARF. In this section we conduct an ablation study with the ML1M dataset. Table 4 shows the results of testing different components in MARF. Firstly, we seek to evaluate the impact of the up-down flexible sampling strategy. We only utilize each user and item rating sequence, sorted by their timestamps to construct $UserBehavior$ and $ItemHistory$. Rather than using the whole user and item sequence profile, we choose the latest session and a random session of each user/item profile where we keep session length $N = 25$ as the input for the sequence extractor layer. Then we put the generated representations into a 2 layer MLP after concatenation. In Table 4, the RS indicates that the model only uses rating feature sequences as input. Compared to using either the latest, or a random sequence, the up-down flexible strategy outperforms the baselines significantly.

To explore the impact of the feedback feature, we choose the popular neural collaborative filtering model [11] as a base model. It uses trained user/item embedding pairs as the input to an MLP prediction layer, and achieves 0.8649 AUC score and 0.3589 log loss on our test set. Then we average each user/item rating in their profile as the overall feedback feature and concatenate them with their embedding as the MLP prediction input, and the performance improves slightly. After changing to our proposed sequence extraction layer with up-down flexible sample strategy to generate user and item embeddings, the performance substantially improves. With additional side features, we get our final reported

[1] https://github.com/doubleblind3372857384/MARF.

Table 4. The performance of different components in MARF

Model	AUC	Log loss
RS-Last Sequence	0.827	0.3924
RS-Random Sequence	0.827	0.3931
RS-up-down flexible sampling	**0.8546**	**0.3649**
Base Model	0.8649	0.3585
Base Model with Feedback	0.8694	0.3523
MARF without Side Features	0.8814	0.3379
MARF	**0.8961**	**0.3158**

results using the MARF model. We can take the following observations from the results in Table 4:

- Flexible up-down sample strategy is necessary for MARF: we can see that the performance drops significantly when it is replaced with the other two methods.
- MARF's user and item representations are superior to randomly initialized embeddings from the NCF model.

3.7 Potential Transferability Analysis

So far, we have described MARF and demonstrated its ability to learn more informative user and item representations. The user representations are learned by combining user features, implicit interaction data from item and feedback signals. Item representations are learned in a similar manner but from item metadata, interaction histories from user, and feedback signals. These rich representations present an opportunity to apply MARF in a transfer learning scenario where two domain datasets have overlapping users or items. For example, commodities could appear in two different platforms, while the *ItemHistories* vary cross different platforms. One platform has a lot of interactions on items called luxury platform while the other has few called sparse platform. Because items in sparse platform have less user interactions which is hard for model to generate informative information, we use the luxury platform datasets to train MARF to get the item representations apply on sparse platform datasets. On the user representations, we can utilise pre-trained item embeddings from luxury platform and user $UserBehavior$ from sparse platform to generate user embeddings.

Table 5. The analysis of transferability of the MARIF without side features

Datasets	AUC score	Log loss
ML100 K	0.8585	0.3759
ML1M	0.8869	0.3159
ML1M to ML100 K	0.7934	0.4409

Accordingly, we conduct the following experiments on ML100 K and ML1 M, which share 1,236 overlapping items—while excluding item metadata and features—to test the transferability of the model. We use ML1M dataset to pretrain the MARF model and then use the ML100 K dataset for evaluation. Without training the model using the ML100 K dataset, we get an AUC score of **0.7934** and a log loss of **0.4409** on the ML100 K test set. Table 5 shows a comparison of the performance between the MARF models trained with and without transferability on two datasets ML100 K, ML1M. It also demonstrates the performance of applying pre-trained embedding from ML1M to ML100 K. Although directly applying pre-trained embeddings from ML1M to ML100 K compromises the performance, it is acceptable compared to the cost of re-training the model.

4 Conclusion and Future Work

In this paper, we proposed a novel deep network method, namely MARF, to model user and item representations. MARF not only enhances the resulting user and item representations, but also leads to a significant improvement on the CTR task. To that end, we designed a flexible up-down sample strategy to sample both representative user and item sequences with feedback, while maintaining the original distribution of user/item rating habits, and also keeps the implicit properties of items/users in different rating categories. Using the projection layer to project the embeddings into the same space and utilizing average sum method to get the final representation of users and items. Their representation becomes more informative than random initialized and easier for CTR prediction task. Last but not least, we show a potential application to transfer learning, if cross domain datasets have either overlapping users or items. In future work, we will try to integrate implicit data which can reflect both user attitudes and item properties.

Acknowledgements. This research is supported by Science Foundation Ireland through the Insight Centre for Data Analytics.

References

1. Aoyama, H.: A study of stratified random sampling. Ann. Inst. Stat. Math. **6**(1), 1–36 (1954)
2. Cheng, H., et al.: Wide & deep learning for recommender systems. In: Karatzoglou, A., et al. (eds.) Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, 15 September 2016, pp. 7–10. ACM (2016)
3. Cheng, W., Shen, Y., Huang, L.: Adaptive factorization network: learning adaptive-order feature interactions. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020, pp. 3609–3616. AAAI Press (2020)

4. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Sen, S., Geyer, W., Freyne, J., Castells, P. (eds.) Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016, pp. 191–198. ACM (2016)
5. Feng, Y., et al.: Deep session interest network for click-through rate prediction. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019, pp. 2301–2307 (2019). ijcai.org
6. Fu, W., Peng, Z., Wang, S., Xu, Y., Li, J.: Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 27 January - 1 February 2019, pp. 94–101. AAAI Press (2019)
7. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning, ICML 2010, Haifa, Israel, 21–24 June 2010, pp. 13–20. Omnipress (2010)
8. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for CTR prediction. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 1725–1731 (2017). ijcai.org
9. He, R., McAuley, J.J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (eds.) Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, 11–15 April 2016, pp. 507–517. ACM (2016)
10. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: Kando, N., Sakai, T., Joho, H., Li, H., de Vries, A.P., White, R.W. (eds.) Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, 7–11 August 2017, pp. 355–364. ACM (2017)
11. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: Barrett, R., Cummings, R., Agichtein, E., Gabrilovich, E. (eds.) Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, 3–7 April 2017, pp. 173–182. ACM (2017)
12. Huang, T., Zhang, Z., Zhang, J.: Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Bogers, T., Said, A., Brusilovsky, P., Tikk, D. (eds.) Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, 16–20 September 2019, pp. 169–177. ACM (2019)
13. Liu, Q., Yu, F., Wu, S., Wang, L.: A convolutional click prediction model. In: Bailey, J., et al. (eds.) Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, 19–23 October 2015, pp. 1743–1746. ACM (2015)
14. Ni, J., Li, J., McAuley, J.J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 188–197. Association for Computational Linguistics (2019)

15. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
16. Song, W., et al.: Autoint: automatic feature interaction learning via self-attentive neural networks. In: Zhu, W., et al. (eds.) Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, 3–7 November 2019, pp. 1161–1170. ACM (2019)
17. Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., Guo, M.: Multi-task feature learning for knowledge graph enhanced recommendation. In: Liu, L., et al. (eds.) The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019, pp. 2000–2010. ACM (2019)
18. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. CoRR abs/1708.05123 (2017)
19. Wang, W., Feng, F., He, X., Zhang, H., Chua, T.: "Click" is not equal to "like": Counterfactual recommendation for mitigating clickbait issue. CoRR abs/2009.09945 (2020)
20. Yen, S.J., Lee, Y.S.: Cluster-based under-sampling approaches for imbalanced data distributions. Expert Syst. Appl. **36**(3), 5718–5727 (2009)
21. Zhou, G., et al.: Deep interest evolution network for click-through rate prediction. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 27 January - 1 February 2019, pp. 5941–5948. AAAI Press (2019)
22. Zhou, G., et al.: Deep interest network for click-through rate prediction. CoRR abs/1706.06978 (2017)