Meta Hierarchical Reinforced Learning to Rank for Recommendation: A Comprehensive Study in MOOCs[⋆]

Yuchen Li[1], Haoyi Xiong[2], Linghe Kong[1](✉), Rui Zhang[1], Dejing Dou[2], and Guihai Chen[1]

[1] Shanghai Jiao Tong University, Shanghai, China
`{yuchenli,linghe.kong,zhang_rui}@sjtu.edu.cn`, `{gchen}@cs.sjtu.edu.cn`
[2] Baidu Inc., Beijing, China
`{xionghaoyi,doudejing}@baidu.com`

Abstract. The rapid development of Massive Open Online Courses (MOOCs) surges the needs of advanced models for personalized online education. Existing solutions successfully recommend MOOCs courses via deep learning models, they however generate weak "course embeddings" with original profiles, which contain noisy and few enrolled courses. On the other hand, existing algorithms provide the recommendation list according to the score of each course while ignoring the personalized demands of learners. To tackle the above challenges, we propose a *Meta hierarchical Reinforced Learning to rank* approach **MRLtr**, which consists of a Meta Hierarchical Reinforcement Learning pre-trained mechanism and a gradient boosting ranking method to provide accurate and personalized MOOCs courses recommendation. Specifically, the end-to-end pre-training mechanism combines a user profile reviser and a meta embedding generator to provide course embedding representation enhancement for the recommendation task. Furthermore, the downstream ranking method adopts a LightGBM-based ranking regressor to promote the order quality with gradient boosting. We deploy **MRLtr** on a real-world MOOCs education platform and evaluate it with a large number of baseline models. The results show that **MRLtr** could achieve $\Delta NDCG_4$= 7.74%∼16.36%, compared to baselines. Also, we conduct a 7-day A/B test using the realistic traffic of Shanghai Jiao Tong University MOOCs, where we can still observe significant improvement in real-world applications. **MRLtr** performs consistently both in online and offline experiments.

Keywords: Online Education · MOOCs Recommendation · Meta Learning · Hierarchical Reinforcement Learning · Learning to Rank.

[⋆] This work was supported in part by National Key R&D Program of China (No. 2021ZD0110303), NSFC grant 62141220, 61972253, U1908212, 72061127001, 62172276, 61972254, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, Open Research Projects of Zhejiang Lab No. 2022NL0AB01. Linghe Kong is the corresponding author.

1 Introduction

With the rapid development of online education, many Massive Open Online Courses (MOOCs) platforms (e.g., Coursera, edX and Udacity) have been built around the world to offer convenient and low-cost opportunities to access high qualified courses from elite universities. The rapid development of MOOCs surges the needs of advanced models and algorithms for personalized course recommendation. Nowadays, deep learning techniques have made significant achievements in many areas, such as computer vision, natural language processing and recommendation system. The MOOCs recommendation can be considered as a sequential recommendation problem. We can formulate the problem as recommending the most probable course to be enrolled by certain user (the user's preference) at time $t+1$, given a set of historical enrolled courses (the user's profile) before time t. To tackle such issue, existing works have proposed various methods to model users' preferences. For example, factored item similarity model (FISM) [1] represents each course as an embedding vector and averages the embedding of all historical courses as the user's preference without capturing the order of the courses. In order to use the order of historical courses, [2] proposes a gated recurrent unit model adding a temporal sequence of the historical courses, whose output is the last vector of the user preference. However, its performance is compromised by assigning all the historical courses with the same weight when calculating the similarity between the target course and the user profile. To distinguish the weights of different courses, two attention-based models (neural attentive item session-based recommendation (NASR) [3] and neural attentive item similarly (NAIS) [4]) are proposed. NAIS and NASR can estimate the attention coefficient of each enrolled course as the importance indicator.

While existing attempts have made significant progress, there still exists three technical challenges as follows. Firstly, existing solutions using deep learning models could fit the original user profiles well, they however contain noises. For instance, there are some enrolled courses whose watching duration is terribly short in a user profile, which represents that the user shows little interest in the courses or enrolls them mistakenly. Once these noisy courses are fed into deep learning models, they will dilute the importance of the contributing courses, which will make the recommendation model performance poorly. Secondly, existing solutions successfully extract features from course materials for recommendation via "course embeddings", they however fail to extract informative features for recommendation when the training data is limited (e.g., cold-start courses). For some courses with many enrolled users, their features will be learned sufficiently. In such case, it will be a higher chance for them to be recommended. On the contrary, for new courses with relatively few enrollments, their special features will be ignored, which leads to lower recommendation probability. Moreover, for cold-start case, when a new course is added into the platform, or the trained model is deployed on a new platform, the recommendation accuracy will be compromised significantly since the embeddings of the new courses can not be represented well. Finally, existing course recommendation systems usually score the courses and provide the recommendation order directly according to the scores [1]. However, this pointwise method only considers a single course at a time in the loss function, which essentially recasts the problem as a regression task. The score of each candidate course is independent without contemplating the potential relationship between the courses. For the MOOCs recommendation, it is necessary to provide more accurate and personalized recommendation order for students.

In order to tackle the above three issues, we propose a three-step approach: (1) *Reinforced User Profiling with Items Filtering*; (2) *End-to-end Pre-training with Meta Enhancing*; (3) *Gradient Boosting with Order Promoting*. Specially, the first step adopts a hierarchical reinforcement learning method to conduct a user profile reviser, which aims to avoid deep learning models overfitting the noise courses. To enhance the representation of course embeddings, a meta embedding generator is proposed, which can not only adapt fast for new courses, but also perform well on few-shot enrolled data. Instead of using the original course embeddings directly, **MRLtr** fuses the features learned from the user profile reviser and the meta embedding generator to provide an end-to-end pre-training for the downstream recommendation models. After the basic recommendation model, the final step replaces the pointwise loss function with a LightGBM [17]-based Learning To Rank (LTR) model, which chooses the listwise loss function to capture the comprehensive user-course relevance for the sake of promoting more accurate and personalized course order for student users.

We conduct extensive offline and online experiments on a real-world MOOCs platform. The results show the effectiveness of **MRLtr** and the consistent performance in the real-world MOOCs platform. To the best of our knowledge, this is the first work to propose an end-to-end pre-trainning mechanism with a LTR promoting method for the MOOCs recommendation task. Our main contributions can be summarized as follows:

- We study the problem of online recommendation in the context of Online Education, where we particularly focus on the technical challenges on embedding representation and order promotion. To the best of our knowledge, this work is the first to investigate course embedding representation with an end-to-end pre-traininng mechanism and order promotion with a LightGBM-based ranking regressor.
- We design and implement **MRLtr**, incorporating the end-to-end pre-training mechanism and the order promotion model in the basic recommendation task. Specifically, **MRLtr** consists of three steps: (1) *Reinforced User Profiling with Item Filtering* that removes the noisy courses with hierarchical reinforcement learning, (2) *End-to-end Pre-training with Meta Enhancing* that adopts a gradient-based meta learning approach to search a better embedding representation and average the embeddings generated from step (1) to feed recommendation models, (3) *Gradient Boosting with Order Promoting* that promotes the course recommendation order via a ranking regressor.
- We deploy **MRLtr** on the Shanghai Jiao Tong University (SJTU) MOOCs and evaluate it using both offline experiments and online A/B tests in comparison with baseline algorithms. The experiment results show that, com-

pared to the state of the art algorithms, **MRLtr** could achieve $\Delta NDCG_4 = 7.74\% \sim 16.36\%$ in offline experiments and significant improvement in online A/B tests under fair comparisons. Extensive ablation studies further confirm the effectiveness of **MRLtr** for the MOOCs recommendation.

2 Background and Formulation

In this section, we introduce the background of the basic MOOCs recommendation problem and formulate the basic MOOCs recommendation model.

2.1 Background

Like all the recommendation tasks, users and items are two basic elements in MOOCs recommendation problem. We use $U = \left\{u_1, \cdots, u_{|U|}\right\}$ to denote the user set and $C = \left\{c_1, \cdots, c_{|C|}\right\}$ to denote the course set of a MOOCs platform. The set of historical enrolled courses is defined as $\mathcal{E}^u = \left(e_1^u, \cdots, e_{t_u}^u\right)$, which also denotes the user profile. We formulate the problem as recommending the most probable course u to be enrolled at $t_u + 1$, given a set of user's historical enrolled courses $\mathcal{E}^u$ before time t.

2.2 Formulations

Like all general recommendation tasks, characterizing the user's preference based on its profile $\mathcal{E}^u$ is critical. We utilize a valued low dimensional embedding vector $\mathbf{p}_t^u$ to represent each historical enrolled course e_t^u. User u's preference is denoted as $\mathbf{q}_u$, which aggregates the embeddings of all historical enrolled courses. An embedding vector $\mathbf{p}_i$ is utilized to represent the target course c_i. The probability p of recommending c_i to u can be represented as

$$p = P\left(y = 1 \mid \mathcal{E}^u, c_i\right) = \sigma_\theta\left(\mathbf{q}_u^T \mathbf{p}_i\right), \tag{1}$$

where $\sigma_\theta(\cdot)$ is the sigmoid function which transforms the input embedding vectors into a probability, θ is its parameters. The key issue for solving the recommendation task is to calculate $\mathbf{q}_u$, i.e., the aggregated embedding. There are some existing methods to obtain $\mathbf{q}_u$. For example, we can average the embeddings of all the historical enrolled courses. However, this method treats all historical enrolled courses equally, which neglects the importance of different courses and cannot represent the real interest of users. Hence, an existing work utilizes the attention mechanism to estimate an attention coefficient a_{it}^u for each course e_t^u [4]. Moreover, there is also a method using attentive recurrent neural network to capture the order of historical courses [3].

In this paper, we adopt the method that parameterizes the attention coefficient a_{it}^u as a function, whose inputs are $\mathbf{p}_t^u$ and $\mathbf{p}_i$. Then the embeddings are calculated based on their attentions as,

$$\mathbf{q}_u = \sum_{t=1}^{t_u} a_{it}^u \mathbf{p}_t^u, a_{it}^u = z\left(\mathbf{p}_t^u, \mathbf{p}_i\right), \tag{2}$$

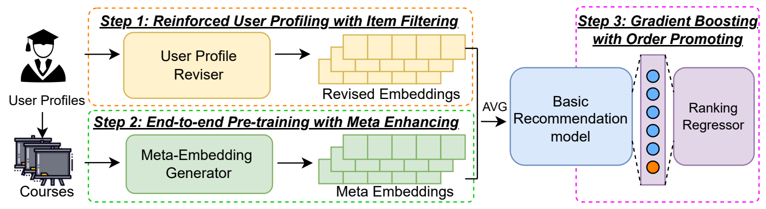

Step 1: Reinforced User Profiling with Item Filtering
User Profile Reviser
Revised Embeddings
User Profiles
Step 2: End-to-end Pre-training with Meta Enhancing
Meta-Embedding Generator
Meta Embeddings
Courses
AVG
Step 3: Gradient Boosting with Order Promoting
Basic Recommendation model
Ranking Regressor

Fig. 1: The pipeline of MRLtr. Firstly, we pre-train the basic recommendation model. Then, we train *Step 1* with the basic recommendation model. Next, we train *Step 2* with the basic recommendation model. Finally, we jointly train all the parts together with the frozen parameters of *Step 2*.

where $z(\cdot)$ can be instantiated by a multi-layer perception on the concatenation or the element-wise product of the two embeddings $\mathbf{p}_t^u$ and $\mathbf{p}_i$.

3 Methodology

In this section, we present the technical details of **MRLtr**. As illustrated in Figure 1, **MRLtr** consists of three steps: (1) *Reinforced User Profiling with Item Filtering*, (2) *End-to-end Pre-training with Meta Enhancing*, and (3) *Gradient Boosting with Order Promoting*. We first introduce *Reinforced User Profiling with Item Filtering* which adpots a hierarchical reinforcement learning algorithm to remove the noisy courses. Second, we propose the *End-to-end Pre-training with Meta Enhancing* which utilizes a gradient-based meta learning algorithm to enhance the representation of course embeddings. Finally, we introduce *Gradient Boosting with Order Promoting* which deploys a LightGBM-based ranking regressor to replace the pointwise loss function for the course recommendation.

3.1 Reinforced User Profiling with Item Filtering

The whole profile revising process can be formulated as a hierarchical Markov Decision Process (MDP), which contains a high-level task and a low-level task. The training process of the profile reviser is shown in Figure 2.

Formulating the item filtering task as a hierarchical MDP An MDP can be represented as a 5-tuple $\langle\mathcal{S},\mathcal{A},\pi,\mathcal{P},\mathcal{R}\rangle$, with $\mathcal{S}$ denoting the state space, $\mathcal{A}$ denoting the action space, π denoting the policy, $\mathcal{P}$ denoting the state transition probability matrix, and $\mathcal{R}$ denoting the reward. Specifically, the agent observes an environment state $s \in \mathcal{S}$, takes an action $a \in \mathcal{A}$ based on a certain policy $\pi(a|s)$, which is the conditional probability density of choosing action a under state s. After applying a, the agent receives a reward $r \in \mathcal{R}$, then the state transfers to s' with probability $p(s'|s,a) \in \mathcal{P}$. The proposed approach reformulates the profile revising task as an MDP $\langle\mathcal{S},\mathcal{A},\pi,\mathcal{P},\mathcal{R}\rangle$, which is given as:
1) state $\mathcal{S}$. We catrgorize $\mathcal{S}$ into the low-level task and the high-level task:

- low-level task: For each historical course e_t^u, the state feature $\mathbf{s}_t^l$ contains four aspects: the effort taken in the course e_t^u, cosine similarity and element-wise product between the embedding vectors of e_t^u and c_i, and the average of the previous features over all historical courses in $\mathcal{E}^u$. Notice that the embedding vector is obtained from a pre-trained basic recommendation model.
- high-level task: For each user profile, the state feature of high-level task $\mathbf{s}^h$ contains: the average cosine similarity and element-wise product between the embedding vectors of all e_t^u in $\mathcal{E}^u$ and c_i, the probability of recommending e_t^u to user u obtained by a basic recommendation model.

2) action $\mathcal{A}$. We categorize $\mathcal{A}$ into the low-level task and the high-level task:

- low-level task: The low-level action $a_t^l \in \{0, 1\}$ for each historical course e_t^u is defined as a binary value to indicate whether to remove it or not.
- high-level task: The high-level action $a^h \in \{0, 1\}$ is defined as a binary value to indicate whether to revise the profile $\mathcal{E}^u$ of user u or not.

3) policy π. We utilize policy networks parameterized by θ^l and θ^h for low-level policy $\pi(\mathbf{s}_t^l, a_t^l|\theta^l)$ and high-level policy $\pi(\mathbf{s}^h, a^h|\theta^h)$, respectively.
4) reward $\mathcal{R}$. We first define a delayed reward for each low-level action as

$$r(a_t^l, \mathbf{s}_t^l) = \begin{cases} \log p(y = 1|\widehat{\mathcal{E}}^u, c_i) - \log p(y = 1|\mathcal{E}^u, c_i), & \text{if} t = t_u \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $\widehat{\mathcal{E}}^u$ is the revised profile. The delayed reward shows the difference between the log-likelihood of recommending c_i after and before the profile is revised.

- high-level task: A delayed reward is used to evaluate the high-level action, $R^h = r(a_t^l, \mathbf{s}_t^l)$. When the high-level task determines to revise the user profile, R^h will be received after the last low-level action is performed.
- low-level task: An internal reward $g(a_t^l, \mathbf{s}_t^l)$ is defined as follows: First, obtain the average cosine similarity between each historical course e_t^u and c_i after and before the profile is revised. Then, calculate the difference between them as $g(a_t^l, \mathbf{s}_t^l)$. Finally, the reward for low-level task is obtained by $R^l = r(a_t^l, \mathbf{s}_t^l) + g(a_t^l, \mathbf{s}_t^l)$.

Algorithm workflow With the formulated hierarchical MDP above, the task of profile reviser is to find a set of optimal parameters $\theta = \{\theta^l, \theta^h\}$ to maximize the expected reward as

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{\tau} p_\theta(\tau) R(\tau), \tag{4}$$

where τ is the sampled sequence (i.e., $\tau = \{\mathbf{s}_1^l, a_1^l, \mathbf{s}_2^l, \cdots, \mathbf{s}_t^l, a_t^l, \cdots \mathbf{s}_{t_u}^l, a_{t_u}^l\}$ for low-level tasks, and $\tau = \{\mathbf{s}^h, a^h\}$ for high-level tasks), $p_\theta(\tau)$ denotes the sampling probability, and $R(\tau)$ denotes the reward for τ.

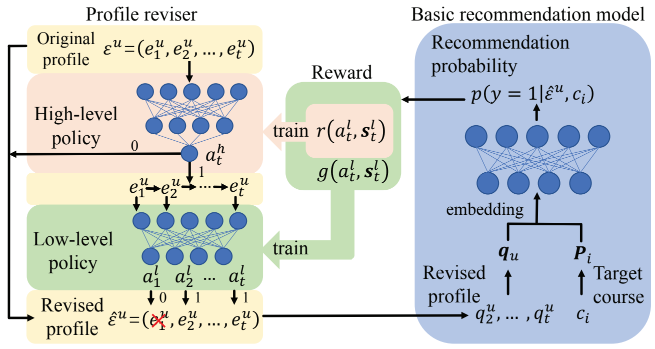

Profile reviser
Original profile
$\mathcal{E}^u=(e_1^u, e_2^u, \ldots, e_t^u)$
High-level policy
a_t^h
0
1
$e_1^u \rightarrow e_2^u \rightarrow \cdots \rightarrow e_t^u$
Low-level policy
a_1^l a_2^l ... a_t^l
Revised profile
$\hat{\mathcal{E}}^u=(e_1^u, e_2^u, \ldots, e_t^u)$
Reward
train
$r(a_t^l, s_t^l)$
$g(a_t^l, s_t^l)$
train
Basic recommendation model
Recommendation probability
$p(y=1|\hat{\mathcal{E}}^u, c_i)$
embedding
q_u
P_i
Revised profile
$q_2^u, \ldots, q_t^u$
Target course
c_i

Fig. 2: The training process of the profile reviser.

We invoke monto-carlo policy gradient method to solve the above profile revising task, which is trained jointly with the basic recommendation model. First, we pre-train the basic recommendation model based on the unrevised user profile. With the pre-trained recommendation model, we then train an initialized profile reviser. Specifically, for each user profile $\mathcal{E}^u$ in one training episode, we first execute the high-level task to get a^h. If the high-level task determines to revise the user profile (i.e., $a^h = 1$), the low-level task will be performed. Then, We sample a low-level sequence τ, and compute $r(a_t^l, \mathbf{s}_t^l)$ and $g(a_t^l, \mathbf{s}_t^l)$. After collecting M trajectories, we update θ^l according to the loss function as

$$L_{\theta^l} = \frac{1}{m} \sum_{m=1}^{M} \sum_{t=1}^{t_u} \nabla_{\theta^l} \log \pi_{\theta^l}(\mathbf{s}_t^m, a_t^m) R^l, \tag{5}$$

while the loss function for updating θ^h is given as

$$L_{\theta^h} = \frac{1}{m} \sum_{m=1}^{M} \nabla_{\theta^h} \log \pi_{\theta^h}(\mathbf{s}^m, a^m) R^h. \tag{6}$$

Finally, based on the revised profile, we train the recommendation model and start another training episode.

3.2 End-to-end Pre-training with Meta Enhancing

In this section, we first propose the meta embedding generator that captures the skill of learning course embeddings through meta-learning. Then, we describe the end-to-end pre-training mechanism with feature fusion.

View of meta learning The essence of MOOCs recommendation is to learn the function $\sigma(\cdot)$ with inputs of the course embedding vectors as shown in (1).

Before learning the function $\sigma(\cdot)$, we need to transform the course into a real-valued vector. From a meta-learning perspective, we reintroduce the notation of the MOOCs recommendation model as

$$p = h\left(c_i\right) = \sigma_\theta\left(\phi_{\mathbf{i}}, c_i\right), \tag{7}$$

where θ is the parameters of function $\sigma(\cdot)$ and $\phi_{\mathbf{i}}$ is the embeddings for course c_i. Actually, $h(\cdot)$ is the same function as $\sigma_\theta(\cdot)$. Then we can recast the MOOCs recommendation as a meta-learning problem via viewing each course as a distinguished task. Specifically, for course $\mathbf{i} = \{1, 2, \cdots\}$, each task $t_{\mathbf{i}}$ corresponds to a specific function $h(\cdot)$. Each task has its own parameters $\phi_{\mathbf{i}}$ and shares the same parameters θ of the basic recommendation model. We aim to train the model to learn how to learn $\phi_{\mathbf{i}}$. This is the analysis that we recast the MOOCs recommendation as meta-learning.

Train the meta embedding generator We choose some courses as the prior tasks with many training samples to pre-train the MOOCs recommendation model. In this way, we can get a well-trained parameter set θ and task-specific parameters $\phi_{\mathbf{i}}$ for each prior task. In order to train the meta embedding generator to learn how to learn course embeddings, we choose a course with few enrollments as the new task $\hat{i}$. Due to the fact that the task-specific parameter $\phi_{\mathbf{i}}$ can not be shared with a new task, we have to train a meta embedding generator to replace its place. For a new course (i.e., a new task $t_{\hat{\mathbf{i}}}$), we use $\phi_{\hat{\mathbf{i}}}^{\text{init}}$ as the initial embedding

$$\phi_{\hat{\mathbf{i}}}^{\text{init}} = f_v(c_{\hat{i}}), \tag{8}$$

where v is the meta-parameter and $f_v(\cdot)$ is the meta embedding generator. The recommendation problem can be shown as

$$\hat{p} = h_{meta}\left(c_{\hat{\mathbf{i}}}\right) = \sigma_\theta\left(\phi_{\mathbf{i}}^{\text{init}}, c_{\hat{\mathbf{i}}}\right). \tag{9}$$

As for each task $t_{\mathbf{i}}$ (i.e., course $c_{\mathbf{i}}$), we can get the training set as $\mathcal{D}_{\mathbf{i}} = \{c_{\mathbf{i}}\}_{j=1}^{N_{\mathbf{i}}}$ with $N_{\mathbf{i}}$ samples. We choose two disjoint mini-batches such as $\mathcal{D}_{\mathbf{i}}^1$ and $\mathcal{D}_{\mathbf{i}}^2$, each with K samples. To simulate the course with relatively few enrollments, we assume the mini-batch size K is far less than half of $N_{\mathbf{i}}$. Then we take a two-step strategy to train the meta embedding generator and get the meta-parameter v: (1) We use $h_{meta}(\cdot)$ on the first mini-batch $\mathcal{D}_{\mathbf{i}}^1$ and get the recommendation result as

$$\hat{p}_1 = h_{meta}\left(c_{\mathbf{i}}\right) = \sigma_\theta\left(\phi_{\mathbf{i}}^{\text{init}}, c_{\mathbf{i}}\right). \tag{10}$$

Meanwhile, we obtain the average loss as

$$l_1 = \frac{1}{K} \sum_{k=1}^{K} \left[-y_1 \log \hat{p}_1 - (1 - y_1) \log\left(1 - \hat{p}_1\right)\right], \tag{11}$$

where k is of the k-th sample from batch $\mathcal{D}_{\mathbf{i}}^1$. (2) We execute the learning process with the second batch of data $\mathcal{D}_{\mathbf{i}}^2$ and then compute the gradient of l_1 and take a step of gradient descent,

$$\phi'_{\mathbf{i}} = \phi_{\mathbf{i}}^{\text{init}} - \alpha \frac{\partial l_1}{\partial \phi_{\mathbf{i}}^{\text{init}}}, \tag{12}$$

where α is the learning rate. Next, we test the trained model on the second batch $\mathcal{D}_{\mathbf{i}}^2$. Specifically, we obtain the recommendation result as

$$\hat{p}_2 = h'_{\text{meta}}(c_{\mathbf{i}}) = \sigma_\theta(\phi'_{\mathbf{i}}, c_{\mathbf{i}}). \tag{13}$$

Meanwhile, we obtain the average loss as

$$l_2 = \frac{1}{K} \sum_{k=1}^{K} \left[-y_2 \log \hat{p}_2 - (1 - y_2) \log(1 - \hat{p}_2)\right]. \tag{14}$$

Next we propose the final loss function l_{final} unified l_1 and l_2 as

$$l_{\text{final}} = a l_1 + b l_2, \tag{15}$$

where $a, b \in [0, 1]$ are the weight coefficients of the loss functions and the sum of a and b is 1 (i.e., $a + b = 1$). The aims for defining the final loss function as (15) is summarized as: (1) For the new courses, we aim to reduce the error of the MOOCs recommendation. Hence, we calculate l_1 in the final loss function different from MAML, which takes l_2 as the final loss function. (2) For the courses with few enrollments (i.e., small number of labeled data), we aim to make them learn fast through gradient updates. Then we calculate the gradient by the chain rule:

$$\frac{\partial l_{\text{final}}}{\partial v} = \frac{\partial l_{\text{final}}}{\partial \phi_{\mathbf{i}}^{\text{init}}} \frac{\partial \phi_{\mathbf{i}}^{\text{init}}}{\partial v} = \frac{\partial l_{\text{final}}}{\partial \phi_{\mathbf{i}}^{\text{init}}} \frac{\partial f_v}{\partial v}, \tag{16}$$

where

$$\frac{\partial l_{\text{final}}}{\partial \phi_{\mathbf{i}}^{\text{init}}} = a \frac{\partial l_1}{\partial \phi_{\mathbf{i}}^{\text{init}}} + b \frac{\partial l_2}{\partial \phi'_{\mathbf{i}}} - ab \frac{\partial l_2}{\partial \phi'_{\mathbf{i}}} \frac{\partial^2 l_1}{\partial \phi_{\mathbf{i}}^{\text{init}^2}}. \tag{17}$$

Eventually, we propose the training algorithm for the meta embedding generator as shown in Alg.1. Specifically, we design a neural network as the meta embedding generator. The inputs of the generator is the course features. In our work, we use the embedding layers of the basic recommendation model in the generator instead of training it from scratch. In order to reduce the number of parameters, we use the parameters of reused layers directly. Then the embeddings from different fields are aggregated by average pooling. Eventually, we use a fully connected layer to get the outputs.

End-to-end pre-training with feature fusion In this section, we propose a simple yet useful design to fuse the embeddings from the meta embedding generator and the user profile reviser. First, we make sure the output embedding from the profile reviser and the meta embedding generator have the same dimension. If a historical course is revised by the profile reviser, the corresponding item of the original output of the profile reviser will be set as 0. Then we average the sum of the corresponding parts of the profile reviser and the meta embedding generator to obtain the new course embedding.

Algorithm 1 Training Meta Embedding Generator

Input: The base model σ_θ, course dataset C, hyper-parameter a,b, learning rate α,β.

```
1:  Randomly initialize $v$;
2:  while not done do
3:      Randomly samples $n$ courses $\{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_n\}$ from $C$;
4:      for $\mathbf{i} \in \{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_n\}$ do
5:          Generate the initial embedding: $\phi_{\mathbf{i}}^{\text{init}} = f_v(c_{\mathbf{i}})$;
6:          Sample mini-batch $\mathcal{D}_{\mathbf{i}}^1$ and $\mathcal{D}_{\mathbf{i}}^2$ each with $K$ samples;
7:          Evaluate loss $l_1$ on $\mathcal{D}_{\mathbf{i}}^1$;
8:          Compute adapted embedding: $\phi'_{\mathbf{i}} = \phi_{\mathbf{i}}^{\text{init}} - \alpha \frac{\partial l_1}{\partial \phi_{\mathbf{i}}^{\text{init}}}$;
9:          Evaluate loss $l_2$ on $\mathcal{D}_{\mathbf{i}}^2$;
10:         Compute loss: $l_{\text{final}} = a l_1 + b l_2$;
11:     end for
12:     $v \leftarrow v - \beta \sum_{\mathbf{i} \in \{\mathbf{i}_1, \ldots, \mathbf{i}_n\}} \frac{\partial l_{\text{final}}}{\partial v}$;
13: end while
```

3.3 Gradient Boosting with Order Promoting

Given the new fusion embedding to the basic recommendation models, **MRLtr** replaces the fully connected layer of the basic recommendation model with a LightGBM-based ranking regressor, which adpots the listwise loss function. We denote a set of user-course pairs with the ranking score as a set of triple such as $\mathcal{T} = \{(u_1, e_1, \boldsymbol{y}_1),\ (u_2, e_2, \boldsymbol{y}_2), (u_3,\ e_3, \boldsymbol{y}_3), \ldots\}$. We aim to gain a LTR scoring function f_s. Therefore, the goal is recast to learn a scoring function f which minimizes the loss as

$$L(f) = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \left(\frac{1}{|e_i|} \sum_{j=1}^{|e_i|} \ell(\boldsymbol{y}_j^i, f_s(u_j^i, e_j^i)) \right), \tag{18}$$

where ℓ represents the loss of the ranking prediction of course e_j^i of user u_i against the ground truth $\boldsymbol{y}_j^i$.

4 Experiments

To demonstrate the effectiveness of **MRLtr**, we present extensive experiments on the SJTU MOOCs platform comparing with a large number of baseline methods. Firstly, we detail the experimental settings. Then, we introduce the results of offline experiments. Finally, the performance of online A/B Test shows the effectiveness of **MRLtr**.

4.1 Experimental Settings

Dataset and Evaluation Methodology We collect the dataset from SJTU MOOCs, a large MOOCs platform with significant number of users. Specifically, we collect a portion of the student users who enrolled courses from September 1st, 2016 to September 1st, 2021. Moreover, we make some standardized processing, such as defining the courses with the same name in different years as the same one. For instance, we unity "Computer Network" from 2016 to 2020 into the same course named "Computer Network". The collected dataset consist of 1,452 courses, 65,649 users, 313,492 users enrolled behaviors, and 23 categories.

To evaluate the performance of **MRLtr**, we use Normalized Discounted Cumulative Gain (NDCG) [16], which has been widely adopted to evaluate the ranking performance. Before introducing NDCG, we first introduce the Discounted Cumulative Gain (DCG) as

$$DCG_N = \sum_{i=1}^{N} \frac{G_i}{\log_2(i+1)}, \tag{19}$$

where G_i denotes the weight assigned to the item's label at position i. A higher G_i indicates that the item is more relevant to the user and correspondingly a better LTR model. However, due to the different lengths of various users, it makes no sense to compare the DCG among them. Then, we utilize the following implementation of NDCG to take a mean across all scores as

$$NDCG_N = \frac{DCG_N}{IDCG_N}, \tag{20}$$

where $IDCG_N$ is the ideal order to normalize the scores. Moreover, the value of NDCG is in the range of $[0,1]$. Similarly, a higher $NDCG_N$ indicates a better LTR model. In this paper, we consider the NDCG of top 10 and 4 ranking results, i.e., $NDCG@10$ and $NDCG@4$.

Experiment setups In this work, all the offline experiments are implemented on a server with 32G Memory, 1 NVIDIA Tesla V100 GPU and 2T Disk. The online experiments are deployed on SJTU MOOCs platform. In order to evaluate the effectiveness of **MRLtr** comprehensively, we adopt seven related models proposed by previous researches as competitors:

- **Bayesian Personalized Ranking (BPR).** This model uses a Bayesian method to optimize the pairwise ranking loss in recommendation tasks.
- **Multi-layer Perception (MLP).** The model use a multi-layer perceptron on a pair of user and course embeddings to learn the probability of recommending the course to the user.
- **Factorization Machine (FM).** FM is a principled approach that can easily incorporate any heuristic features.
- **Factored Item Similarity Model (FISM).** FISM is an item-to-item collaborative filtering (CF) algorithm which recommends courses via averaging embedding of all enrolled courses and embedding of the target courses.
- **Gated Recurrent Unit (GRU).** GRU is a gated recurrent unit model that receives a sequence of historical courses as input, then output the last hidden vector as the representation of a user's preference.

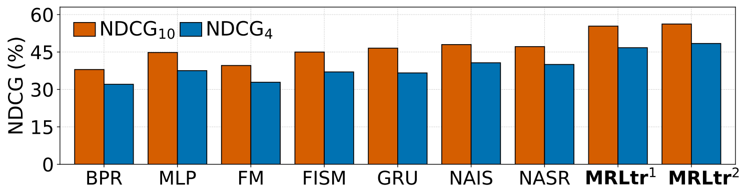


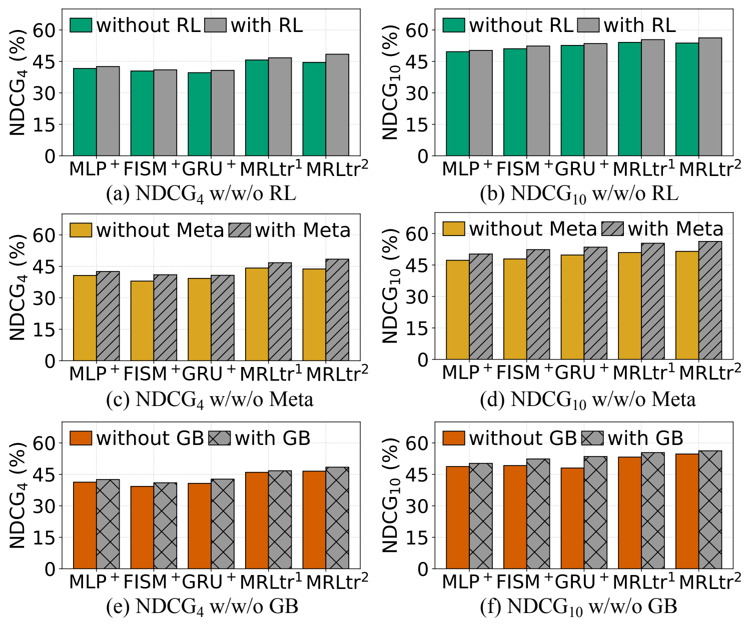
Fig. 3: Offline comparative results of MRLtr and baselines on $\Delta NDCG_{10}$ and $\Delta NDCG_4$. We use **MRLtr*** and **MRLtr**** to represent the **MRLtr**+NAIS and **MRLtr**+NASR, respectively.

- **Neural Attentive Item Similarity (NAIS).** NAIS is a collaborative filtering algorithm which utilizes an attention mechanism to distinguish the weights of different historical enrolled courses.
- **Neural Attentive Session-based Recommendation (NASR).** NASR is an improved model GRU model that estimates attention coefficients for historical enrolled courses based on the corresponding hidden vector outputs.

4.2 Offline Experimental Results

To comprehensively evaluate **MRLtr**, we conduct experiments to answer the following research questions:
RQ1: How does **MRLtr** perform compared with state-of-the-art models for MOOCs recommendation tasks?
RQ2: Is the *Reinforced User Profiling with Item Filtering* in **MRLtr** necessary for improving performance?
RQ3: Is the *End-to-end Pre-training with Meta Enhancing* in **MRLtr** vital for improving performance?
RQ4: How does the *Gradient Boosting with Order Promoting* impact the performance of **MRLtr**?

Comparative Results: RQ1 In Figure 3, we report the offline performance of **MRLtr** compared with other baselines on $NDCG_{10}$ and $NDCG_4$. In order to represent the combination of **MRLtr** with two basic recommendation models briefly, we use **MRLtr**^{1} and **MRLtr**^{2} to represent the combination of **MRLtr** with NAIS and NASR, respectively. Intuitively, we could see that **MRLtr** gains the best performance compared with other baselines on both two metrics. Specifically, **MRLtr**^{2} improves the performance of the baseline models from 7.74% to 16.36% on $NDCG_4$ and from 8.24% to 18.24% on $NDCG_{10}$. Moreover, there are some findings from the comparative experiments. Firstly, all the user-to-item based collaborative models (i.e., BPR, MLP and FM) show poor performance since most of the users in our dataset enrolled a few courses, and the embeddings can not be extracted from the sparse data. Secondly, item-to-item based collaborative filtering models (i.e., FISM and GRU) perform better than user-to-item based collaborative models, but they still perform worse than the attention



(a) NDCG$_4$ w/w/o RL (b) NDCG$_{10}$ w/w/o RL

(c) NDCG$_4$ w/w/o Meta (d) NDCG$_{10}$ w/w/o Meta

(e) NDCG$_4$ w/w/o GB (f) NDCG$_{10}$ w/w/o GB

Fig. 4: Ablation studies of *Reinforced User Profiling with Item Filtering* (RL), *End-to-end Pre-training with Meta Enhancing* (Meta) and *Gradient Boosting with Order Promoting* (GB) for MRLtr on $NDCG_4$ and $NDCG_{10}$. To briefly represent the models, we use MLP$^+$, FISM$^+$, GRU$^+$, **MRLtr**^{1} and **MRLtr**^{2} to represent the combinations of **MRLtr** with MLP, FISM, GRU, NAIS, and NASR, respectively. Moreover, "w/w/o" is the abbreviation of "with or without".

models. Because FISM and GRU treat all historical courses equally. As for NAIS and NASR, we find that they perform better than all the above collaborative filtering models, as they can distinguish the importance of different courses via attention mechanism.

Ablation Study: RQ2 We conduct a series of ablation studies to prove the effectiveness of *Reinforced User Profiling with Item Filtering*, *End-to-end Pre-training with Meta Enhancing* and *Gradient Boosting with Order Promoting* for **MRLtr**. Figure 4 (a) and (b) illustrates that all the models with the *Reinforced User Profiling with Item Filtering* based user profile reviser could obtain better performance compared with the models without the user profile reviser. As shown in Figure 4 (a), *Reinforced User Profiling with Item Filtering* achieves the improvement with 3.91% for **MRLtr**^{2} on $NDCG_4$, which is the largest improvement in this study.

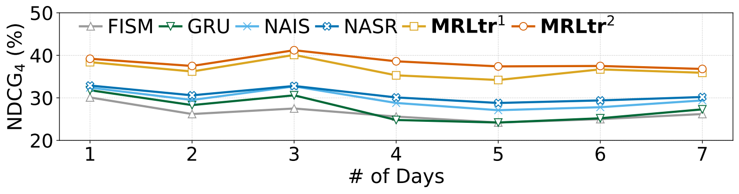


Fig. 5: Online comparative performance on $NDCG_4$ of MRLtr and baselines for 7 days. (t-test with $p < 0.05$ over the baseline).

Ablation Study: RQ3 In order to demonstrate the usefulness of *End-to-end Pre-training with Meta Enhancing*, we conduct a serious of ablation studies. As shown in Figure 4 (c) and (d), the chosen models with *End-to-end Pre-training with Meta Enhancing* (Meta) based the meta embedding generator perform better than the models without the meta embedding generator. Specifically, the meta embedding generator obtains the largest margin with 4.72% on $NDCG_{10}$. These phenomenons prove that *End-to-end Pre-training with Meta Enhancing* could enhance the representation of course embdeddings. There are many users enrolled few courses in the dataset. The meta embedding generator can not only adapt fast for new courses, but also perform well on few-shot enrolled data.

Ablation Study: RQ4 In Figure 4 (e) and (f), we report the ablation study of *Gradient Boosting with Order Promoting* of **MRLtr**. Similarity, all the models with the *Gradient Boosting with Order Promoting* based ranking regressor could obtain better performance compared with the models without the ranking regressor. As depicted in Figure 4 (f), *Gradient Boosting with Order Promoting* achieves the improvement of 5.48% for GRU$^+$ on $NDCG_{10}$. These results demonstrate that the listwise-based ranking regressor performs better. The listwise function treats the whole document list as a sample and directly optimizes the evaluation metrics, such as the utilized metric $NDCG$ in this work.

4.3 Online Experimental Results

To demonstrate the effectiveness of **MRLtr**, we conduct a series of online A/B tests with real-world web traffics and compare it with the baseline models on SJTU MOOCs platform. According to the offline experimental results, we conduct the online experiments with full real-world web traffic, which last for 7 days. Figure 5 illustrates the comparison of **MRLtr** with the baselines on $\Delta NDCG_4$. Firstly, **MRLtr** could boost the performance compared with the online base system in all days, which demonstrates that **MRLtr** is practical for improving the performance of SJTU MOOCs. Furthermore, we can find that **MRLtr** achieves significant improvements on the real-world MOOCs platform. Specifically, we observe that **MRLtr** outperforms the online base model (FISM) by a large margin on $\Delta NDCG_4$ with 13.8% relative improvement, which reveals the effectiveness of **MRLtr**. Finally, we observe that **MRLtr** performs stably in all days, which demonstrates the soundness and usefulness of our proposed model. Basically, the online performance is consistent with offline experiment results.

5 Related Work

CF has been widely used in sequential recommendation problems, where each user-item interaction data naturally forms a sequence for being associated with timestamp information. For example, bayesian personalized ranking [5], matrix factorization [6] and factorization machine [7] are all user-to-item based CF methods. However, the performances of the above models are limited when data is sparse. By contrast, the item-to-item CF models can handle the above problem. [1] is proposed to calculate the item similarity via dot product of item embeddings. To retrieve the main preference in the sequential data, attention mechanism was proposed in NASR [3] and NAIS [4]. Moreover, RNN [8] and GRU [2] are used to capture the temporal factor of the user-item iteration data.

Recently, some researches attempt to adapt meta-learning algorithm to solve issues. Meta-learning aims to adapt a trained model to new tasks quickly and effectively by using the prior experience learned from the related tasks [9]. For example, Model-Agnostic Meta-Learning (MAML) are proposed to solve the cold-start problem [10]. Recently, motivated by the aforementioned benefits of meta-learning, it has been invoked into the recommendation tasks [11]. Moreover, λOpt [12] is proposed to optimize regularization hyper-parameters based on validation performance.

According to the loss function, we could categorize the LTR models into three families: pointwise [13], pairwise [14]and listwise [15]. The listwise model treats the whole document list as a sample and directly optimizes the evaluation metrics, such as the utilized metric in this work, i.e., NDCG.

6 Conclusion

In this paper, we design, implement and deploy a novel MOOCs recommendation approach **MRLtr** on a real-world MOOCs platform to address the problems which contains course data noises, weak representation for few enrolled courses and the poor recommendation order. **MRLtr** contains three steps: (1) *Reinforced User Profiling with Item Filtering* that removes the noisy courses with hierarchical reinforcement learning, (2) *End-to-end pre-training with Meta Enhancing* adopts a gradient-based meta learning approach to search a better embedding representation, and (3) *Gradient Boosting with Order Promoting* promotes the course recommendation order via a LightGBM-based ranking regressor. To verify the effectiveness of **MRLtr**, we conduct extensive offline and online experiments compared with a large number of baseline methods. Offline experiment results show that **MRLtr** could achieve significant gain over baselines on $NDCG_4$ compared with other baselines. Furthermore, **MRLtr** significantly boosts the online MOOCs recommendation performance in real-world applications, which is consistent with the offline results.

References

1. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 659–667 (2013)
2. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. CoRR abs/1511.06939 (2016)
3. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 1419–1428 (2017)
4. He, X., He, Z., Song, J., Liu, Z., Jiang, Y.G., Chua, T.S.: Nais: Neural attentive item similarity model for recommendation. IEEE Transactions on Knowledge and Data Engineering 30(12), 2354–2366 (2018)
5. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
6. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
7. Rendle, S.: Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST) 3(3), 1–22 (2012)
8. Tan, Y.K., Xu, X., Liu, Y.: Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st workshop on deep learning for recommender systems. pp. 17–22 (2016)
9. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017)
10. Bharadhwaj, H.: Meta-learning for user cold-start recommendation. 2019 International Joint Conference on Neural Networks (IJCNN) pp. 1–8 (2019)
11. Ren, Y., Chi, C., Jintao, Z.: A survey of personalized recommendation algorithm selection based on meta-learning. In: The International Conference on Cyber Security Intelligence and Analytics. pp. 1383–1388. Springer (2019)
12. Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J.G., Wang, Y.: λopt: Learn to regularize recommender models in finer levels. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 978–986 (2019)
13. Li, P., Wu, Q., Burges, C.: Mcrank: Learning to rank using multiple classification and gradient boosting. In: Advances in Neural Information Processing Systems. pp. 65–72 (2008)
14. Zheng, Z., Chen, K., Sun, G., Zha, H.: A regression framework for learning ranking functions using relative relevance judgments. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 287–294 (2007)
15. Taylor, M., Guiver, J., Robertson, S., Minka, T.: Softrank: optimizing non-smooth rank metrics. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. pp. 77–86 (2008)
16. J¨arvelin, K., Kek¨al¨ainen, J.: Ir evaluation methods for retrieving highly relevant documents. In: ACM SIGIR Forum. vol. 51, pp. 243–250. ACM New York, NY, USA (2017)
17. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems 30 (2017)