Hierarchical Reinforcement Learning for Integrated Recommendation

Ruobing Xie*, **Shaoliang Zhang***, **Rui Wang, Feng Xia, Leyu Lin**
WeChat Search Application Department, Tencent, China
ruobingxie@tencent.com

Abstract

Integrated recommendation aims to jointly recommend heterogeneous items in the main feed from different sources via multiple channels, which needs to capture user preferences on both item and channel levels. It has been widely used in practical systems by billions of users, while few works concentrate on the integrated recommendation systematically. In this work, we propose a novel Hierarchical reinforcement learning framework for integrated recommendation (HRL-Rec), which divides the integrated recommendation into two tasks to recommend channels and items sequentially. The low-level agent is a channel selector, which generates a personalized channel list. The high-level agent is an item recommender, which recommends specific items from heterogeneous channels under the channel constraints. We design various rewards for both recommendation accuracy and diversity, and propose four losses for fast and stable model convergence. We also conduct an online exploration for sufficient training. In experiments, we conduct extensive offline and online experiments on a billion-level real-world dataset to show the effectiveness of HRL-Rec. HRL-Rec has also been deployed on WeChat Top Stories, affecting millions of users. The source codes are released in https://github.com/modriczhang/HRL-Rec.

1 Introduction

Personalized recommendation systems can help users to get various types of information including news (Zheng et al. 2018), videos (Hidasi et al. 2016), products (Sun et al. 2019) and tags (Liu et al. 2020c). **Integrated recommendation** is proposed to *simultaneously* recommend these *heterogeneous* items from different sources (i.e., *channels*) in a *single* recommendation system, which has been widely deployed by real-world content providers such like *WeChat Top Stories*, *Google* and *Baidu Feed* (Xie et al. 2020b). Fig. 1 gives a classical architecture of a real-world integrated recommendation. It first combines heterogeneous items from different channels (e.g., article, video and news channels), and then jointly ranks them in the main feed. Integrated recommendation is more convenient for information acquisition. Moreover, it greatly increases users' choices on different types/sources of information to meet user diverse preferences on both item and channel levels, making the systems more attractive and friendly.

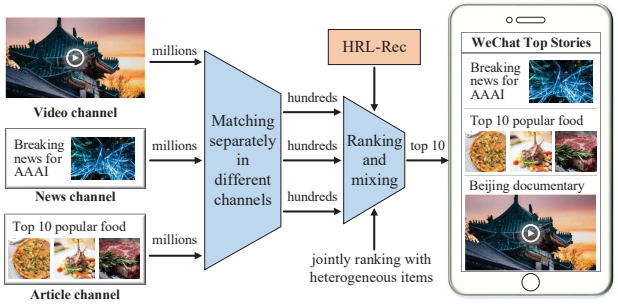

Figure 1: A real-world integrated recommendation system.

A good integrated recommendation system should provide user-interested, diversified and stable results from heterogeneous channels. However, integrated recommendation faces more challenges than conventional homogeneous recommendation: (1) heterogeneous items from multiple channels usually have different features and ranking strategies. It is extremely difficult to compare items in different channels for joint ranking. (2) Users have personalized preferences on both item and channel levels. The coarse-grained user preference on channels also has essential impacts on recommendation. (3) Real-world integrated recommendation highly values the online model stability. A small disturbance in one channel (e.g., data or model updating) may result in serious influences on the overall performances. Although integrated recommendation is widely deployed in practice, there are still few works focusing on these challenges systematically. Currently, most real-world integrated recommendation systems jointly rank heterogeneous items with CTR-oriented objectives and rule-based strategies. However, merely relying on CTR may lead to homogenization and biases in both items and channels (e.g., videos usually have higher CTR compared to articles), which will harm the channel diversity and user experience eventually. Simple empirical rule-based strategies will inevitably reduce the personalization.

To address these challenges and improve the overall performance of integrated recommendation, we propose a novel **Hierarchical reinforcement learning framework for integrated recommendation (HRL-Rec)**, which captures user

* indicates equal contribution. Ruobing Xie is the corresponding author (ruobingxie@tencent.com).

preferences on both heterogeneous items and channels. Precisely, HRL-Rec models the integrated recommendation as a list-wise recommendation, which contains two reinforcement learning (RL) agents. The low-level agent is a **channel selector**, which generates a personalized channel list according to user channel-level preferences. In contrast, the high-level agent works as an **item recommender**, which recommends specific heterogeneous items under the channel constraints given by the channel selector. We design a new set of rewards to measure both accuracy and diversity in this task, and combine low- and high- level HRL losses, supervised loss and unsupervised similarity loss for fast and stable training. We also conduct an online exploration as an unbiased simulator. The advantages of HRL-Rec locate in three aspects: (1) the trial-and-error procedure, multiple losses and the unbiased online exploration of RL help HRL-Rec to effectively find the optimum solutions. (2) HRL-Rec considers multiple rewards to measure the accuracy, diversity and novelty. It could relieve the homogenization issue in CTR-based models. (3) The hierarchical structure decouples channel selector and multi-channel item recommenders, which alleviates the disturbances caused by small changes in a single channel, making online system more stable. It also brings flexibility in customization for each channel.

In experiments, we conduct extensive offline and online evaluations for HRL-Rec with competitive baselines on a real-world system WeChat Top Stories. The significant improvements in both accuracy and diversity confirm the effectiveness of HRL-Rec in practice. Moreover, We further give sufficient online and offline ablation tests, parameter analyses and stability analyses to better understand our model. The main contributions are concluded as follows:

- We systematically explore the integrated recommendation task, and propose a novel HRL-Rec model. To the best of our knowledge, we are the first to bring hierarchical reinforcement learning in integrated recommendation.
- We propose a novel HRL framework specially for integrated recommendation, which designs a new set of rewards for multiple objectives, combines HRL, supervised and similarity losses for fast and stable training, and conducts an online exploration for unbiased simulation.
- The offline and online evaluations, model analyses and ablation tests verify the effectiveness and stability. Currently, HRL-Rec has been deployed on WeChat Top Stories, affecting millions of users.

2 Related Works

Recommendation System. Logistic regression (LR) (Peng, Lee, and Ingersoll 2002) and Factorization machine (FM) (Rendle 2010) are classical methods for recommendation. Wide&Deep (Cheng et al. 2016) combines LR with DNN components. DeepFM (Guo et al. 2017), NFM (He and Chua 2017) and AFM (Xiao et al. 2017) combine FM with DNN or attention layers. AutoInt (Song et al. 2019), AFN (Cheng, Shen, and Huang 2020) and AutoFIS (Liu et al. 2020a) are proposed to better extract feature interactions. BERT4Rec (Sun et al. 2019) and DFN (Xie et al. 2020a) are also proposed for modeling user behavior sequences. In HRL-Rec, we use GRU and self-attention for feature modeling in state encoders, and compare with some strong baselines.

Reinforcement Learning (RL). Policy gradient (PG) and Deep Q-network (DQN) are representative models for policy-based and value-based RL approaches (Mnih et al. 2015; Chen et al. 2019a). Double (Van Hasselt, Guez, and Silver 2016) and Dueling (Wang et al. 2016) strategies are effective for DQN, while A3C (Mnih et al. 2016) and DDPG (Lillicrap et al. 2016) successfully combine advantages in both policy-based and value-based approaches. In this work, we train HRL-Rec with DDPG.

RL in Recommendation. Recently, Zheng et al. (2018) applies DQN to model long-term reward inferred from user activeness. Zhao et al. (2018) conducts Actor-Critic in the page-wise recommendation with neural reward simulators. Adversarial training is also deployed to generate sufficient training instances (Chen et al. 2019b), while Wang et al. (2018) combines RL with supervised methods. Moreover, (Chen et al. 2019a) improves PG with the top-k off-policy correction, while (Ie et al. 2019) proposes the slate-based Q-learning to model long-term value. Both methods have achieved great improvements in online systems.

In contrast, there are only a few works that consider hierarchical RL (HRL) in recommendation. (Zhang et al. 2019) introduces HRL to alleviate noises in user behaviors, which divides the MOOC recommendation into a profile reviser and a basic recommender. (Zhao et al. 2020) uses HRL to highlight high-quality but sparse rewards of conversion in E-commerce, where the high-level agent is regarded as an inductor of the low-level agent. Differing from these models, HRL-Rec focuses on the integrated recommendation task, which aims to jointly consider user preferences on both items and channels to rank heterogeneous items. To the best of our knowledge, we are the first to bring HRL in integrated recommendation. Existing HRL models in recommendation (Zhang et al. 2019; Zhao et al. 2020) cannot be directly used in our task, for they are specially designed for other purposes with customized information (e.g., conversion).

3 Methodology

We propose HRL-Rec for integrated recommendation. As in Fig. 1, multiple *channels* (i.e. data sources) are combined for joint heterogeneous ranking. In this work, we focus on four channels including *news*, *article*, *long video* and *short video*.

3.1 Problem Formulation and Model Overview

Integrated recommendation aims to *simultaneously* recommend heterogeneous items from *different channels* in a *single* system. Precisely, we define the integrated recommendation as a **list-wise** recommendation task. The inputs are heterogeneous items from different channels, and the output is a recommended list (i.e., top 10 items) containing heterogeneous items from different channels. We use RL to address this task. The agent (i.e., model) selects an *action* (i.e., recommending a channel or an item), and the environment (i.e., users) gives *rewards* (e.g., click, dwell time (Yi et al. 2014), diversity, novelty). The agent then learns from the rewards and updates the states. HRL-Rec aims to maximize the expected multi-aspect rewards of the overall list.

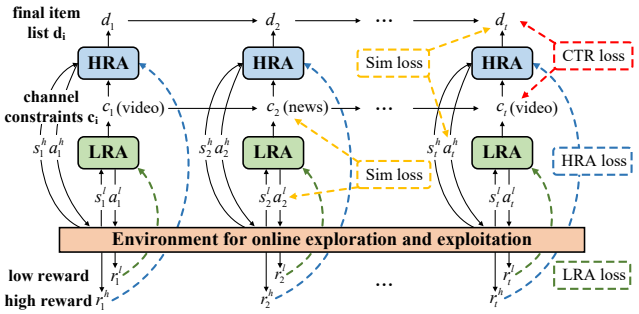

Figure 2: The overall architecture of HRL-Rec.

Fig 2 shows the overall architecture of HRL-Rec, which mainly consists of a channel selector and an item recommender. The **channel selector** is regarded as the low-level RL agent (LRA), which gives a coarse-grained channel recommendation with a channel list. The **item recommender** is viewed as the high-level RL agent (HRA), which recommends specific heterogeneous items under the channel constraints. We use the DDPG (Mnih et al. 2016) to model both HRA and LRA, and define the key notions as follows:

- **Low-level state** s^l: s^l contains information of user profiles, user historical behaviors, recommendation contexts, and recent channel impressions (impression indicates the item/channel is exposed/shown to a user).
- **Low-level action** a^l: the low-level action is generating a channel. a^l is viewed as the recommended channel.
- **Low-level reward** r^l: the low-level reward r^l is measured by the click times of the corresponding channel.
- **High-level state** s^h: s^h contains information of user profiles, user historical behaviors, recommendation contexts, recently item impression, and channel constraints.
- **High-level action** a^h: the high-level action is generating an item. a^h is viewed as the recommended item.
- **High-level reward** r^h: the high-level reward derives from four factors, including item click times, dwell time, list-level diversity and item novelty, measuring recommendation accuracy, diversity and novelty in item level.
- **Discount factor** γ: the discount factor $\gamma \in [0, 1]$ measures the importance of future rewards to the current state.

When recommending the t-th item in the final item list, LRA first observes the low-level state s^l_t and gives an action a^l_t to recommend a channel c_t in the t-th position. Second, HRA the high-level state s^h_t and generates the high-level action a^h_t indicating the specific item d_t in channel c_t. Third, the environment receives c_t and d_t, and sends both low-level and high-level rewards r^l_t and r^h_t back to the agents. Finally, both states are updated with corresponding state transitions.

The two-step HRL can (1) learn user preferences on both items and channels for joint recommendation with heterogeneous items (verified in Sec. 4.4), (2) consider multi-aspect rewards that measure both accuracy and diversity in online scenarios (verified in Sec. 4.5), and (3) decouple channel selector and each item recommender, which assures the stability in online systems (verified in Sec. 4.8). In the following sections, we further introduce the details and advantages of all components in HRL-Rec (see Sec. 4.5, 4.6 and 4.7).

3.2 Channel Selector (LRA)

The LRA is a channel selector to generate channel lists with the classical Actor-Critic framework (Lillicrap et al. 2016).

Low-level State Encoder The t-th low-level state s^l_t aims to capture (1) user historical impressed channels, (2) user profiles, and (3) previous $t-1$ items HRL-Rec has already recommended in the list. Therefore, different from existing HRL models (Zhao et al. 2020), we use the **impression** behavior sequence instead of the click sequence as inputs. Precisely, when predicting the t-th channel, we use the m most recent impressed behaviors before the t-th position to form the input feature sequence, noted as $seq^l_t = \{\mathbf{f}^l_1, \cdots, \mathbf{f}^l_m\}$. The former $m - t + 1$ behaviors are the recent historical channel impressions, while the latter $t - 1$ behaviors are the channels already predicted by LRA in the current list.

The i-th feature embedding $\mathbf{f}^l_i$ is generated by four groups of raw features: (a) user long-term profiles, (b) recommendation contexts, (c) current channel features, and (d) cumulative channel features in recent impressions. Following (Song et al. 2019), we divide these raw feature groups into n feature fields to form the input feature matrix $\hat{\mathbf{F}}^l_i \in \mathbb{R}^{n \times d_h}$. We conduct a multi-head self-attention to model feature interactions between n fields of the i-th behavior, and concatenate n interacted field features to get $\mathbf{f}^l_i$ as follows:

$$\mathbf{f}^l_i = \text{Concat}(\text{Transformer}(\hat{\mathbf{F}}^l_i)), \quad \mathbf{f}^l_i \in \mathbb{R}^{nd_h}. \quad (1)$$

The recent impressed items and channels are essential to represent the state in list-wise recommendation. Therefore, we use RNN with Gated recurrent units (GRU) (Hidasi et al. 2016) as the sequential state encoder, and set the initial state as user profiles. We conduct GRU on the input sequence seq^l_t to represent the t-th low-level state embedding $\mathbf{s}^l_t$ as:

$$\mathbf{s}^l_t = \text{GRU}(seq^l_t) = \text{GRU}(\mathbf{f}^l_1, \cdots, \mathbf{f}^l_{m-t+1}, \cdots, \mathbf{f}^l_m). \quad (2)$$

$\text{GRU}(seq^l_t)$ is the last hidden state in GRU. We propose a novel state encoder specially for integrated recommendation to capture useful information at the channel level. The state embedding $\mathbf{s}^l_t$ learns list-level information from two aspects, including the hidden states of sequence-based model and the cumulative channel features functioned as a short path.

Low-level Actor The low-level Actor generates actions to represent channels according to the low-level state embeddings. Specifically, we feed $\mathbf{s}^l_t$ into a fully connected layer to generate the t-th low-level action embedding $\mathbf{a}^l_t$ as:

$$\mathbf{a}^l_t = \tanh(\mathbf{W}^l_a \cdot \mathbf{s}^l_t + \mathbf{b}^l_a). \quad (3)$$

$\mathbf{a}^l_t$ is viewed as the virtual channel embedding recommended by LRA. We calculate the cosine similarities between $\mathbf{a}^l_t$ and all item candidates in multiple channels, and directly regard the channel of the most similar item as the predicted channel c_t. Note that we choose item embeddings instead of channel embeddings to calculate similarity with $\mathbf{a}^l_t$, since the fine-grained item embeddings are more precise to represent user preferences on channels than the coarse-grained channel embeddings. c_t is viewed as the channel constraint in HRA.

Low-level Critic The low-level Critic aims to calculate the Q value which represents the total reward after generating a channel. The low-level Q value evaluates the expected low-level return of the overall list, which is formalized as:

$$Q^l(s_t^l, a_t^l) = \mathbb{E}_{s_{t+1}^l, r_t^l \sim E}[r_t^l + \gamma Q^l(s_{t+1}^l, a_{t+1}^l)]. \quad (4)$$

r_t^l is the low-level reward from the t-th channel, which indicates the number of the channel c_t being clicked in the t-th position. In this case, LRA can capture user's coarse-grained preferences in channel level at each position. $\gamma \in [0, 1]$ is a discount factor. This discount factor considers the influence brought by position bias in practice, which enables LRA to pay more attention to the items with higher ranks.

Specifically, we use a fully connected layer to estimate the Q^l as q^l. In the t-th position, the Critic inputs the low-level state $\mathbf{s}_t^l$ and the virtual action $\mathbf{a}_t^l$ to predict the current Q as:

$$q^l(s_t^l, a_t^l) = \mathrm{ReLU}(\mathbf{w}_{c1}^{l\top} \cdot \mathbf{s}_t^l + \mathbf{w}_{c2}^{l\top} \cdot \mathbf{a}_t^l + b_c^l), \quad (5)$$

where $\mathbf{w}_{c1}^l$ and $\mathbf{w}_{c2}^l$ are weighting vectors and b_c^l is the bias. The Q value given by Critic is then used to guide the update in Actor and Critic, which will be introduced in Sec. 3.4.

3.3 Item Recommender (HRA)

The HRA aims to recommend a heterogeneous item list under the channel constraints generated by LRA.

High-level State Encoder The state encoder in HRA focuses more on fine-grained item-level features and impressions instead of channel-level features. Similar to LRA, the input feature sequence seq_t^h is also formed by the m-most recent impressed behaviors as $seq_t^h = \{\mathbf{f}_1^h, \cdots, \mathbf{f}_m^h\}$. $\mathbf{f}_i^h$ is built from the similar feature groups in $\mathbf{f}_i^l$, where the current and cumulative channel features are replaced with the corresponding item features. Taking the similar feature field matrix $\hat{\mathbf{F}}_i^h$ as input, we calculate $\mathbf{f}_i^h$ via the Transformer used in Eq. (1) as $\mathbf{f}_i^h = \mathrm{Concat}(\mathrm{Transformer}(\hat{\mathbf{F}}_i^h))$ for feature interaction. Finally, we conduct another GRU model to learn the high-level state embedding $\mathbf{s}_t^h = \mathrm{GRU}(seq_t^h)$ from recent item impression sequences.

High-level Actor The high-level Actor generates actions to recommend heterogeneous items in different channels under the hard channel constraint c_t given by LRA. In this case, the t-th item could ***only*** be selected from the c_t channel. We also conduct a fully connected layer as:

$$\mathbf{a}_t^h = \tanh(\mathbf{W}_a^h \cdot \mathbf{s}_t^h + \mathbf{b}_a^h). \quad (6)$$

$\mathbf{a}_t^h$ is the virtual action embedding given by HRA. We then retrieve the most similar item d_t of $\mathbf{a}_t^h$ in channel c_t with cosine similarity, and regard d_t as the recommended item.

High-level Critic The high-level Critic aims to measure the total reward when Actor predicts an item. The high-level Q value $Q^h(s_t^h, a_t^h)$ of expected return is calculated similar as Eq. (4) with the high-level reward r_t^h. To jointly consider recommendation **accuracy**, **diversity** and **novelty**, r_t^h contains four rewards related to different core indicators in list-wise integrated recommendation: (1) r_t^{click} indicates the number of the item d_t being clicked by users, which directly optimizes the click-related objective. (2) r_t^{time} measures the dwell time (i.e., user's reading time) of d_t (Yi et al. 2014), which can reveal user's true satisfaction. (3) r_t^{diver} measures the recommendation diversity with the increments of deduplicated tags/categories brought by the t-th item compared to existing $t-1$ items in the current list. (4) r_t^{novel} evaluates the novelty with the numbers of new tags/categories in the t-th item compared to those already in user's long-term content profiles. The former two rewards focus on user short-term click-related performances, while the latter two rewards focus on diversity and novelty, which help to improve user long-term experience. The final aggregated high-level reward r_t^h is the combination of four rewards as:

$$r_t^h = \sum_{i=1}^{4} (\mathbf{r}_i^t + b_i^r)^{\lambda_i^{c_t}}, \mathbf{r}^t = \{r_t^{click}, r_t^{time}, r_t^{diver}, r_t^{novel}\}. \quad (7)$$

b_i^r is the bias and $\lambda_i^{c_t}$ is the weight. Heterogeneous channels usually emphasise different rewards in practice. Hence, we set channel-specific reward weights $\lambda_i^{c_t}$ that vary with channel constraint c_t. We conduct a fully connected layer to estimate the Q value $q^h(s_t^h, a_t^h)$ with $\mathbf{a}_t^h$ and $\mathbf{s}_t^h$ as:

$$q^h(s_t^h, a_t^h) = \mathrm{ReLU}(\mathbf{w}_{c1}^{h\top} \cdot \mathbf{s}_t^h + \mathbf{w}_{c2}^{h\top} \cdot \mathbf{a}_t^h + b_c^h), \quad (8)$$

where $\mathbf{w}_{c1}^h$ and $\mathbf{w}_{c2}^h$ are weighting vectors and b_c^h is the bias.

3.4 Multi-aspect Objectives in HRL-Rec

To ensure the rapid and stable convergence of HRL-Rec, we design a multi-aspect objective jointly modeling four losses, including (1) LRA loss, (2) HRA loss, (3) CTR-oriented supervised loss, and (4) similarity loss. In training, we utilize DDPG (Lillicrap et al. 2016) to train the Actors and Critics in both agents with off-policy strategy. We also adopt the double strategy (Van Hasselt, Guez, and Silver 2016).

LRA loss. In LRA, we adopt the classical mean squared loss (MSE) to train the low-level Critic as follows:

$$\begin{aligned} L(\theta^l) &= \mathbb{E}_{s_t^l, r_t^l \sim E}[(y_t^l - Q^l(s_t^l, a_t^l|\theta^l))^2], \\ y_t^l &= r^l(s_t^l, a_t^l) + \gamma Q^l(s_{t+1}^l, \pi^l(s_{t+1}^l|\phi^{l'})|\theta^{l'}). \end{aligned} \quad (9)$$

y_t^l is the target Q value at t that consists of the current reward r^l and the future Q value from the target network. θ^l and $\theta^{l'}$ are the parameters of the online and target low-level Critics. θ^l is updated during training, while $\theta^{l'}$ is the previous experience parameter set fixed during optimization. $\pi^l(s_{t+1}^l|\phi^{l'})$ represents the target policy of the low-level Actor.

Similar to DDPG, the Q value generated by Critic is used to update the Actor with policy gradient. We maximize the overall list-wise expected return, aiming to generate actions with higher Q values. HRL-Rec learns the parameter ϕ^l of our low-level Actor by the loss $L(\phi^l)$ as follows:

$$L(\phi^l) = \mathbb{E}_{s^l \sim E, a^l = \mu^l(s^l|\phi^l)}[-Q^l(s^l, a^l|\theta^l)]. \quad (10)$$

$a^l = \mu^l(s^l|\phi^l)$ indicates the virtual action embedding in Eq. (3) generated by the deterministic policy of low-level Actor. Finally, the LRA loss L_l is aggregated as follows:

$$L_l = L(\theta^l) + \beta L(\phi^l). \quad (11)$$

β is a hyper-parameter of weight empirically set as 1.

HRA loss. In HRA, we build the MSE loss for the high-level Critic similarly to Eq. (9) as follows:

$$L(\theta^h) = \mathbb{E}_{s_t^h, r_t^h \sim E}[(y_t^h - Q^h(s_t^h, a_t^h|\theta^h))^2],$$
$$y_t^h = r^h(s_t^h, a_t^h) + \gamma Q^h(s_{t+1}^h, \pi^h(s_{t+1}^h|\phi^{h'})|\theta^{h'}). \tag{12}$$

θ^h and $\theta^{h'}$ are parameters of the online and target network in high-level Critic. The loss of high-level Actor is also as:

$$L(\phi^h) = \mathbb{E}_{s^h \sim E, a^h = \mu^h(s^h|\phi^h)}[-Q^h(s^h, a^h|\theta^h)]. \tag{13}$$

μ^h is the policy of high-level Actor in Eq. (6). Finally, we add both losses in Actor and Critic to form the HRA loss:

$$L_h = L(\theta^h) + \beta L(\phi^h). \tag{14}$$

In model training, to enhance the robustness of HRL, we further add *Gaussian noises* $\mathbf{a}_n \sim N(0, \sigma^2)$ to both action embeddings in Eq. (3) and Eq. (6). The Gaussian noise brings randomness in training, and thus could improve the generalization ability and robustness of our model.

CTR-oriented supervised loss. In practice, it is challenging for RL to fast converge since it usually has large gradient variances (especially for hierarchical RL). HRL models may lost in optimization due to the natural sparsity of click behaviors compared to the tremendous search space. Therefore, we bring in supervised information to jointly guide the HRL-Rec in training. We design a CTR-oriented supervised loss L_c with the predicted high-level and low-level actors a and the corresponding real item $\hat{d}$ displayed to user u as:

$$L_c = -\sum_{\hat{d} \in C_u} \log(f(a, \hat{d})) - \sum_{\hat{d} \notin C_u} \log(1 - f(a, \hat{d})). \tag{15}$$

$\hat{d} \in C_u$ indicates $\hat{d}$ is clicked by user u. $f(a, \hat{d})$ is the predicted click probability with a and $\hat{d}$, represented as:

$$f(a, \hat{d}) = \text{sigmoid}(\mathbf{w}_f^\top \cdot \text{concat}(\mathbf{a}, \hat{\mathbf{d}}) + b_f). \tag{16}$$

$\mathbf{w}_f$ is a weighting vector and b_f is a bias. L_c aims to build implicit relations between the virtual actions and their corresponding real clicked items. It directly optimizes the high-level and low-level Actors via click-based supervised information, which could make the training more robust and accelerate model convergence in HRL training.

Similarity loss. We further design a similarity loss L_s in both LRA and HRA with the virtual action $\mathbf{a}$ and its most similar item $\mathbf{d}$ given by Actors as follows:

$$L_s = -\sum_{(a,d)} \text{cosine\_similarity}(\mathbf{a}, \mathbf{d}). \tag{17}$$

L_s attempts to make actions (i.e., virtual items and channels) generated by Actors closer to their most similar real items, which forces Actors to generate more realistic actions.

Finally, we combine all four losses to get the final loss as:

$$L = \lambda_l L_l + \lambda_h L_h + \lambda_c L_c + \lambda_s L_s. \tag{18}$$

We empirically set the loss weights $\lambda_l : \lambda_h : \lambda_c : \lambda_s = 5 : 5 : 1 : 1$ according to model performances. For fast and stable convergence, we update Critics with a higher frequency (usually 5 to 30 times) than Actors, and also conduct a distributed learning with asynchronous gradient optimization (Mnih et al. 2016). HRL-Rec can be trained within 3 hours.

3.5 Online Exploration

Most RL-based methods mainly rely on simulators for exploration (Zhao et al. 2020). However, simulator-based RL in recommendation seriously suffers from noises and overfitting, for there are gaps between simulated and real feedbacks. Differing from simulator-based RL, we adopt the ε-greedy strategy (Mnih et al. 2015) in online system as an **online exploration**, which randomly generates "surprising" items from the input item candidates in HRL-Rec (i.e., top 200 items in different channels). These items are impressed to real-world users, and thus have unbiased user feedbacks to be collected as exploration for offline training.

The feasibility and advantages of using online exploration instead of simulators are as follows: (1) our online system is widely used by millions of users. The RL with online exploration and additional supervised training assure the sufficient training of HRL-Rec. (2) Simulators inevitably bring in inaccurate and biased rewards, while all explorations in HRL-Rec are evaluated by real users. (3) The qualities of online exploration are acceptable, since the "randomly" selected items are top 200 items ranked by the previous matching module. The online exploration will not harm user experience in the long term. In online, we regard the most similar item of the predicted action in HRA as the final result.

4 Experiments

4.1 Datasets

Since HRL-Rec relies on online exploration and there is no large-scale dataset for our setting, we build a new dataset IRec-4B from a real-world integrated recommendation system named WeChat Top Stories. We randomly select 22.5 million users and collect nearly 141 million sessions from their logs. These sessions contain million-level deduplicated items from *news*, *article*, *long video* and *short video* channels, generating 3.8 billion impression instances. We split these instances into a train set and a test set using the chronological order. All data are preprocessed via data masking to protect user privacy. Table 1 shows the statistics of IRec-4B.

#user	#session	#click	#impression
22,527,380	140,982,284	355,019,232	3,770,955,760

Table 1: Statistics of the IRec-4B dataset.

4.2 Competitors

We implement several competitive models as baselines. The competitors include conventional LR (Peng, Lee, and Ingersoll 2002) and FM (Rendle 2010) models. For deep models, we implement Wide&Deep (Cheng et al. 2016), NFM (He and Chua 2017), AFM (Xiao et al. 2017), DeepFM (Guo et al. 2017) and AutoInt (Song et al. 2019), which are widely confirmed in industry. Note that we cannot directly use existing HRL models in recommendation (Zhang et al. 2019; Zhao et al. 2020) for they are designed for different tasks with customized feedbacks. All baselines can use the same features also used in HRL-Rec if necessary, and are trained under the classical CTR-based cross entropy loss.

4.3 Experimental Settings

HRL-Rec takes top 200 items in each channel as inputs and output top 10 heterogeneous items. The maximum length of input sequence is 50 for both agents. The dimensions of the aggregated feature embeddings and the item/channel embeddings are 128 and 32. We utilize a 4-head self-attention, and set the discount factor as $\gamma = 0.3$. The channel-specific high-level reward weights $\lambda_i^{c_t}$ are set flexibly and empirically according to the specific online requirements. In training, we use Adam for optimization with the batch size set as 256. We conduct a grid search for parameter selection. All models share the same features and experimental settings.

4.4 Offline Evaluation

Evaluation Protocols Integrated recommendation *jointly* provides heterogeneous items. Hence, we evaluate all models on the CTR prediction task in IRec-4B. We use the classical Area Under Curve (AUC) for offline evaluation. Following (Yan et al. 2014), we also bring in RelaImpr to measure the relative improvements over the base model (i.e., LR).

model	AUC	RelaImpr
LR (Peng, Lee, and Ingersoll 2002)	0.7311	0.00%
FM (Rendle 2010)	0.7585	11.86%
NFM (He and Chua 2017)	0.7620	13.37%
AFM (Xiao et al. 2017)	0.7686	16.23%
Wide&Deep (Cheng et al. 2016)	0.7801	21.20%
DeepFM (Guo et al. 2017)	0.7819	21.98%
AutoInt (Song et al. 2019)	0.7837	22.76%
HRL-Rec (ours)	**0.8097**	**34.01%**

Table 2: Results of offline evaluation on IRec-4B.

Experimental Results In Table 2 we can find that:

(1) HRL-Rec significantly outperforms all baselines with the significance level $\alpha = 0.01$. The improvements verify that HRL-Rec can capture both item-level and channel-level user preferences to rank heterogeneous items.

(2) The improvements of HRL-Rec mainly derive from three aspects: (a) the hierarchical RL structure decouples channel selector and item recommender, guiding the model to learn both item-level and channel-level user preferences. (b) The trial-and-error of RL, online exploration and multi-aspect loss help HRL-Rec to effectively find the optimum selections. (c) The high-level reward considers recommendation accuracy, diversity and novelty from different aspects, which improves both short-term and long-term experiences.

4.5 Online A/B Tests

Evaluation Protocols We evaluate HRL-Rec on WeChat Top Stories. Precisely, we deploy HRL-Rec with different RL methods in the ranking module with other modules unchanged. The online base model is a supervised LR model with rule-based strategies for channels. In online evaluation, we focus on three representative metrics including Click-through-rate (CTR), average click number per capita (ACN) and average watched tag per capita (AWT) to model recommendation accuracy and diversity. The online evaluation could be viewed as an online ablation test.

RL models	CTR	ACN
DQN (LR)	+4.17%	+3.72%
DQN (GRU)	+5.27%	+4.77%
Double-Dueling-DQN	+5.40%	+5.41%
DDPG	+5.80%	+7.82%
DDPG (hierarchical)	+6.07%	+10.43%
HRL-Rec (final)	**+6.34%**	**+11.67%**

Table 3: Online A/B test on WeChat Top Stories.

Experimental Results We conduct the online A/B test for 5 days. Table 3 shows the improvement percentages over the base model. We can observe that:

(1) HRL-Rec significantly outperforms the online baseline and other RL versions on both CTR and ACN metrics. CTR measures the point-wise accuracy. In contrast, ACN indicates both accuracy and user activeness in online list-wise recommendation, which we care more about. In HRL-Rec, we utilize the click number as the dominating reward in both agents, since we hope users to click more items. Hence, we achieve more significant improvements in ACN.

(2) Comparing with different RL models from DQN (LR) to HRL-Rec, we can find that: (a) DQN performs better than the LR/rule-based baseline. (b) GRU outperforms LR as the state encoder in RL models. (c) Double and Dueling strategies are effective in DQN for our task. (d) Actor-Critic based models such as DDPG could help model optimization and relieve large gradient variance issue. (e) The hierarchical RL structure can decouple the channel selector and item recommender, which is beneficial for integrated recommendation. (f) The supervised loss L_c further improves the performance.

(3) We further analyze the improvements in diversity with average watched tag per capita (AWT). We add the diversity and novelty rewards r^{diver} and r^{novel} to the last two RL models. Hence, the AWT gets 1.98% increase from DDPG to DDPG (hierarchical) and HRL-Rec. A good recommendation diversity could improve user's long-term experience.

4.6 Ablation Tests

We conduct ablation tests to verify the effectiveness of different components in HRL-Rec. Table 4 shows the results of various ablation settings. We find that: (1) both self-attention (for feature field interaction) and GRU (for sequence encoding) are essential in two state encoders. Here we replace GRU and self-attention with fully-connected layers and average pooling as ablation settings. (2) $\gamma = 0$ indicates that HRL-Rec will not consider any future rewards at each position of the list-wise recommendation. It implies the effectiveness of future rewards in list-wise recommendation. (3) Both L_s and L_c are indispensable in our model. It is because that the similarity loss helps Actors to generate more realistic items and channels, while the supervised loss directly optimizes HRL-Rec with CTR. Besides the AUC improvements, the more important necessity of these two losses is

that they can help to improve the stability of the training, ensuring the rapid and stable convergence in HRL.

Task	AUC	RelaImpr
HRL-Rec	**0.8097**	**34.01%**
– GRU	0.7879	24.58%
– self-attention	0.8065	32.63%
– future rewards ($\gamma = 0$)	0.7967	28.39%
– similarity loss L_s	0.8045	31.76%
– supervised click loss L_c	0.8032	31.20%

Table 4: Ablation tests on IRec-4B dataset.

4.7 Model Analyses

Analysis on Discount Factor γ We first evaluate HRL-Rec with different γ to analyze the impacts of discount factors on the performances. Fig. 3 (a) shows the AUC with different γ: (1) in general, the AUC increases first and then decreases as γ grows from 0 to 1. HRL-Rec reaches the best performance when $\gamma = 0.3$. It is natural since the future rewards in HRL-Rec reflect the accuracy or diversity factors of items with lower ranks in the recommended list, which should be considered less important when measuring current states. Small γ also appears in other recommendation tasks (Zheng et al. 2018; Xin et al. 2020; Liu et al. 2020b). (2) $\gamma = 0$ indicates that HRL-Rec merely considers instant rewards brought by the current action, ignoring any future rewards (e.g., list-wise clicks and diversity). The improvement verifies the significance of future rewards.

Analysis on Input Sequence Length We further explore the model performance with different maximum input sequence length of seq^l and seq^h in Fig. 3 (b). We find that: (1) the AUC first increases and then decreases with the sequence length from 5 to 100. HRL-Rec achieves the best performance when the length equals 50. (2) The AUC increases sharply at the beginning, which indicates that the *impressed* items in user historical behaviors are helpful to model the current state. The AUC decreases slowly from 50 to 100, which reveals the side effects of considering too long-term behaviors in list-wise recommendation.

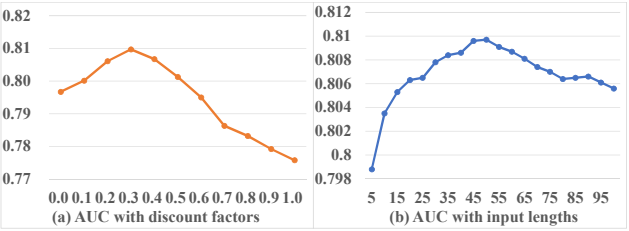

Figure 3: Results of parameter analyses.

4.8 Online Model Stability

In real-world integrated recommendation, online model stability is an essential factor that severely impacts user experience. However, heterogeneous items from multiple channels bring in diversity as well as instability. Any change of data or model in a certain channel may lead to a huge disturbance in the overall results, which will harm the user experience.

We assume that a practical and robust integrated recommendation system should have a **stable channel proportion** with daily model and data updates. Hence, we use the *stability* of *channel proportion* to evaluate the online model stability. In Fig. 4, we draw the trend chart of the proportions of different channels in the total. We compare HRL-Rec with DQN on *video* and *news* channels, where both models need daily updates. Since the channel proportions varies regularly with different days and times, we compare with *the* ***same time*** *(e.g., Sat. 16:00) in* ***two adjacent weeks***. We observe that the channel proportion trend in DQN differs in adjacent weeks. The max and average relative changes can even reach 18.0% and 11.7% in *video* channel. In contrast, our HRL-Rec is much more stable after model update. Its max and average relative changes are only 4.5% and 1.4% in *video* channel. It is because that (1) HRL-Rec has successfully learned user preferences on channels (which are stable), and (2) HRL-Rec decouples channel selector and item recommender with different parameters and rewards. Hence, it could smooth the channel proportion jitters caused by disturbances from biased data or unbalanced model, and thus can enhance user experience and increase user stickiness.

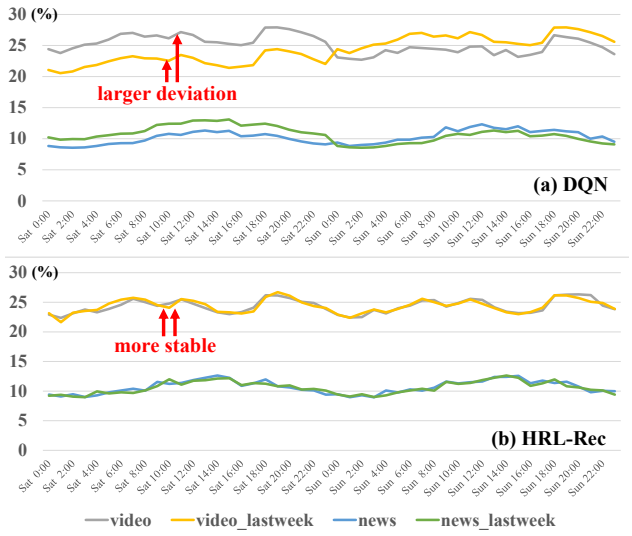

Figure 4: The trend chart of different channel proportions in two adjacent weeks with daily model and data updates.

5 Conclusion and Future Work

In this work, we highlight integrated recommendation and propose a new HRL-Rec, which is divided into channel selection and item recommendation with multiple rewards and losses. We conduct extensive offline and online experiments and achieve significant improvements. HRL-Rec has been deployed on WeChat Top Stories, affecting millions of users.

In the future, we will explore more effective rewards and complicated HRL frameworks. We will further use the off-policy correction in online exploration for better exploration.

References

Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. H. 2019a. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of WSDM*.

Chen, X.; Li, S.; Li, H.; Jiang, S.; Qi, Y.; and Song, L. 2019b. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System. In *Proceedings of ICML*.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the DLRS workshop*.

Cheng, W.; Shen, Y.; and Huang, L. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *Proceedings of AAAI*.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of IJCAI*.

He, X.; and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of SIGIR*.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of ICLR*.

Ie, E.; Jain, V.; Wang, J.; Navrekar, S.; Agarwal, R.; Wu, R.; Cheng, H.-T.; Lustman, M.; Gatto, V.; Covington, P.; et al. 2019. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. In *Proceedings of IJCAI*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *Proceedings of ICLR*.

Liu, B.; Zhu, C.; Li, G.; Zhang, W.; Lai, J.; Tang, R.; He, X.; Li, Z.; and Yu, Y. 2020a. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction .

Liu, F.; Guo, H.; Li, X.; Tang, R.; Ye, Y.; and He, X. 2020b. End-to-End Deep Reinforcement Learning based Recommendation with Supervised Embedding. In *Proceedings of WSDM*.

Liu, Q.; Xie, R.; Chen, L.; Liu, S.; Tu, K.; Cui, P.; Zhang, B.; and Lin, L. 2020c. Graph Neural Network for Tag Ranking in Tag-enhanced Video Recommendation. In *Proceedings of CIKM*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of ICML*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* .

Peng, C.-Y. J.; Lee, K. L.; and Ingersoll, G. M. 2002. An introduction to logistic regression analysis and reporting. *The journal of educational research* .

Rendle, S. 2010. Factorization machines. In *Proceedings of ICDM*.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of CIKM*.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of CIKM*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of AAAI*.

Wang, L.; Zhang, W.; He, X.; and Zha, H. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of KDD*.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *Proceeding of ICML*.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of IJCAI*.

Xie, R.; Ling, C.; Wang, Y.; Wang, R.; Xia, F.; and Lin, L. 2020a. Deep Feedback Network for Recommendation. In *Proceedings of IJCAI*.

Xie, R.; Qiu, Z.; Rao, J.; Liu, Y.; Zhang, B.; and Lin, L. 2020b. Internal and Contextual Attention Network for Cold-start Multi-channel Matching in Recommendation. In *Proceedings of IJCAI*.

Xin, X.; Karatzoglou, A.; Arapakis, I.; and Jose, J. M. 2020. Self-Supervised Reinforcement Learning forRecommender Systems. In *Proceedings of SIGIR*.

Yan, L.; Li, W.-J.; Xue, G.-R.; and Han, D. 2014. Coupled group lasso for web-scale ctr prediction in display advertising. In *Proceedings of ICML*.

Yi, X.; Hong, L.; Zhong, E.; Liu, N. N.; and Rajan, S. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of RecSys*.

Zhang, J.; Hao, B.; Chen, B.; Li, C.; Chen, H.; and Sund, J. 2019. Hierarchical Reinforcement Learning for Course Recommendation in MOOCs. In *Proceedings of AAAI*.

Zhao, D.; Zhang, L.; Zhang, B.; Zheng, L.; Bao, Y.; and Yan, W. 2020. MaHRL: Multi-goals Abstraction Based Deep Hierarchical Reinforcement Learning for Recommendations. In *Proceedings of SIGIR*.

Zhao, X.; Xia, L.; Zhang, L.; Ding, Z.; Yin, D.; and Tang, J. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of RecSys*.

Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N. J.; Xie, X.; and Li, Z. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of WWW*.