Aggregating E-commerce Search Results from Heterogeneous Sources via Hierarchical Reinforcement Learning

Ryuichi Takanobu*
Institute for AI, State Key Lab of Intelligent Technology & Systems, DCST, Tsinghua University
Beijing, China
gxly15@mails.tsinghua.edu.cn

Tao Zhuang*
Alibaba Group
Hangzhou, China
zhuangtao.zt@alibaba-inc.com

Minlie Huang†
Institute for AI, State Key Lab of Intelligent Technology & Systems, DCST, Tsinghua University
Beijing, China
aihuang@tsinghua.edu.cn

Jun Feng‡
State Grid Zhejiang Electric Power Co., LTD
Hangzhou, China

Haihong Tang
Alibaba Group
Hangzhou, China

Bo Zheng
Alibaba Group
Hangzhou, China

ABSTRACT

In this paper, we investigate the task of aggregating search results from heterogeneous sources in an E-commerce environment. First, unlike traditional aggregated web search that merely presents multi-sourced results in the first page, this new task may present aggregated results in all pages and has to dynamically decide which source should be presented in the current page. Second, as pointed out by many existing studies, it is not trivial to rank items from heterogeneous sources because the relevance scores from different source systems are not directly comparable. To address these two issues, we decompose the task into two subtasks in a hierarchical structure: a high-level task for *source selection* where we model the sequential patterns of user behaviors onto aggregated results in different pages so as to understand user intents and select the relevant sources properly; and a low-level task for *item presentation* where we formulate a slot filling process to sequentially present the items instead of giving each item a relevance score when deciding the presentation order of heterogeneous items. Since both subtasks can be naturally formulated as sequential decision problems and learn from the future user feedback on search results, we build our model with hierarchical reinforcement learning. Extensive experiments demonstrate that our model obtains remarkable improvements in search performance metrics, and achieves a higher user satisfaction.

CCS CONCEPTS

• **Information systems** → **Combination, fusion and federated search**; *Online shopping*; • **Computing methodologies** → **Reinforcement learning**; *Learning from implicit feedback*.

*Both authors contributed equally to this research.
†Corresponding author: Minlie Huang.
‡Participated in this work while at Tsinghua University.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313455

KEYWORDS

aggregated search, vertical, user feedback, hierarchical reinforcement learning

ACM Reference Format:
Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. 2019. Aggregating E-commerce Search Results from Heterogeneous Sources via Hierarchical Reinforcement Learning. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313455

1 INTRODUCTION

The process of aggregating search results from heterogeneous sources is usually referred to as *aggregated search* [17]. Different from a meta-search engine or search federation which assumes that all distributed sources contain homogeneous content and applies the same ranking function to each source for result fusion [1], an aggregated search system is expected to support **heterogeneous information seeking** from different sources. When a user issues a query, the search system should integrate relevant results from each specialized search system, called *vertical*, into one Search Engine Result Page (SERP).

Existing research on search result aggregation has focused on web search [4, 11, 21], while in this paper, we study the task in E-commerce search with one of the largest E-commerce search services in the world: *Taobao.com*. Fig. 1 illustrates that traditional E-commerce search can be augmented with *blog posts* that share purchasing experiences, or *topic groups* that cluster products with the same brand or from the same shop to facilitate finding similar products. As a matter of fact, user study in Taobao Search shows that mixing products with topic groups and blog posts in a SERP can greatly improve the user experiences of product search and online shopping. In this study, we aggregate two other verticals in product search, namely the topic and blog verticals. Given a query, these verticals return topic groups and blog posts respectively.

In web search [4, 11, 21], result aggregation is performed only once at the first page for each query. While in E-commerce search, aggregation is performed multiple times for a query, one for each

Figure 1: An aggregated search example for the query "dress", where the topic group from the *topic* vertical is shown in the 2nd position, and the blog post from the *blog* vertical in the 5th position.

page. The aim of page-wise aggregation in E-commerce search is to keep diversity to attract and retain diverse customers. This is a major difference from previous web search aggregation in which a single decision is made only for the first page.

In this paper, we decompose search result aggregation in E-commerce search into two subtasks: *source selection* and *item presentation*. *Source selection* decides whether to present the search results of a certain source type in the current page, which depends on the user behaviors onto the items that have been already presented in previous pages. For example, if a user issued the query "sport shoes", and clicked several products with the same brand, presenting a topic group with the same brand in the next page may be a good decision. Then the validity of search results can be examined afterwards by analyzing the long-term gains like click through rate (CTR). So this process is a sequence of decisions and can be modeled with reinforcement learning (RL) [32], which helps better understand user intents to present more relevant aggregated search results.

The other subtask, *item presentation*, is to decide the presentation order of the items from heterogeneous sources, which requires to estimate the relevance scores of all items in a unified way. However, items from different sources have different relevance estimation models [2, 16, 28], thereby making the scores not comparable between different sources. This is termed the *relevance ranking* issue. To avoid this, we formulate the item presentation task as a *slot filling* problem where the training signal comes directly from the user feedback. In this formulation, each display position in a page is regarded as a slot, and the slots in a page will be filled sequentially. At each position, an *item presenter* is trained to choose the most relevant item from the candidate sources, and then fills in the slot with the selected item. Once again, the slot filling process is consistent with the sequential nature of user behaviors, which can be viewed as a sequential decision problem and handled by RL.

Given the above decomposition, we propose a hierarchical reinforcement learning (**HRL**) model [5] consisting of two components: a source selector that decides which sources should be selected at the current page, and an item presenter that decides the presentation order of the items selected from the selected sources in a page. The model fully utilizes the sequential characteristics of user behaviors across different pages to decide both the sources and items to be presented. The hierarchical framework enables the model to utilize the immediate user feedback in the current page as well as the future user feedback on the entire search session.

To summarize, our main contributions are as follows:

- We propose a novel search result aggregation method that formulates a semi-Markov decision process which is composed of a high-level policy for *source selection* and a low-level policy for *item presentation*.
- We present aggregated search results for each page during source selection and capture the sequential patterns of user behaviors onto aggregated items in different pages.
- We formulate the item presentation subtask as a slot filling problem in order to avoid the relevance ranking issue on displaying heterogeneous items.

Results demonstrate that our proposed model can improve user experience and make significant advancement on marketing metrics like CTR and GMV over other baselines.

2 PRELIMINARIES

2.1 Reinforcement Learning

Reinforcement learning (RL) [32] is a learning paradigm that an agent learns from the interactions with the environment through sequential exploration and exploitation. Generally, RL follows a Markov Decision Process (MDP) formulation.

Let an MDP be defined as $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma >$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ denotes the transition function to generate the next state from current state-action pair, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ denotes the reward function and $\gamma \in [0, 1]$ is an discount factor . Given an MDP, a trajectory $\tau = s_0, a_0, r_0, s_1, \cdots$ that starts from state s_0 is sampled from a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ that specifies the action. The aim of RL is to find an optimal policy π^* which maximizes the expected cumulative reward (return) $R_t = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$ at

each time step t. For an agent following a policy π, the Q-value of the state-action pair (s, a) is defined as $Q_\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a, \pi] = \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[r + \gamma \mathbb{E}_{a' \sim \pi(s')}[Q_\pi(s', a')]|s, a]$, which measures the average discounted long-term rewards. In the practice of training, the action is often selected by an $\epsilon - greedy$ policy that follows the greedy policy $a = \arg\max_{a'} Q(s, a')$ with probability $1 - \epsilon$ for exploitation and selects a random action with probability ϵ for exploration, in order to collect desired transitions for learning.

With the advance of deep neural networks, deep reinforcement learning (DRL) has been widely used in various search tasks in recent two years [12, 25, 41].

2.2 Options Framework

Options framework [33] extends the usual MDP so that it involves temporal abstractions over the action space in the context of HRL. At each step, the HRL agent chooses either a "one-step" action (primitive action) or a "multi-step" action (option). An option $o = (\pi, \mathcal{I}, \beta) \in O$ defines a policy π over actions (either primitive or option), includes an initiation set $\mathcal{I} \subset \mathcal{S}$ that option o is available *iff* state $s \in \mathcal{I}$, and can be terminated according to a stochastic function $\beta : \mathcal{S} \rightarrow [0, 1]$. An MDP endowed with a set of options is extended to a Semi-Markov Decision Process (SMDP). Given an SMDP, a trajectory τ is sampled from a policy μ over options. Each option o_k launches a new subtask where a sub-trajectory τ_k is sampled from the corresponding policy π over actions. The agent cannot select next option (possibly in a recursive way) until current option terminates according to the termination function β.

Recently, several studies have demonstrated that combining DRL with predefined subgoals delivers promising results in challenging environments like Atari [18], Minecraft [35], relation extraction [34] and task-oriented dialogues [26].

3 RELATED WORK

In recent years, search engines are able to search heterogeneous sources that contain different types of contents (e.g. news, maps, videos, images). Studies have shown that the aggregated view helps expose the content in other sources, thus offering the chance for the user to explore different sources in search[7, 10]. Existing aggregated search systems, which merge all heterogeneous results from various search services in one SERP, mostly adopt a pipeline architecture as illustrated in Fig. 2. In general, the architecture contains two key components, one is source selection which decides the vertical sources from which items should be ranked, and the other is item presentation which decides the presentation order of the heterogeneous items.

3.1 Vertical Selection

The first subtask of aggregated search is vertical selection, also called source selection. The goal of vertical selection is to decide which verticals are to be presented in a SERP, e.g. whether it is necessary to present the videos of "violin" when a user types query "violin". Several works [20, 37, 44] have found that participants rated the system negatively when irrelevant vertical results were presented in an aggregated SERP. It is thus essential to evaluate the relevance of a source to a query.

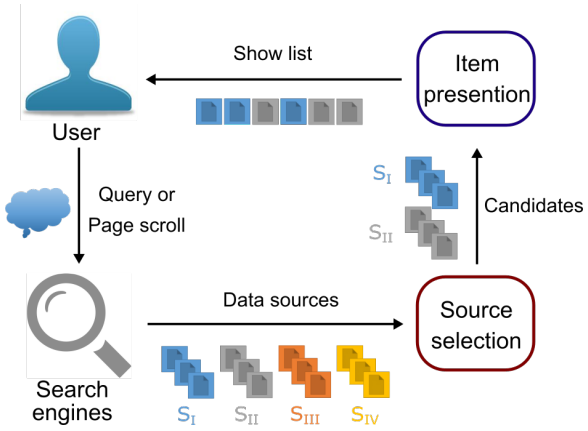

Figure 2: Task flow of search aggregation on heterogeneous sources.

Most existing approaches use a binary classifier for each vertical to make a decision on whether to present a vertical result in a page [15, 19, 36]. From this perspective, each classifier can adopt a different feature representation and focus on the features that are uniquely predictive for its corresponding vertical. Human judges are employed to generate ground-truth labels for the verticals in some approaches [4, 11]. Other approaches [16, 28] leverage user search logs and assign a binary label based on user behaviors.

As previous studies focused on web search, vertical selection is only launched one time when a user issues a query. For instance, the *video* vertical will only appear once in the first SERP for query "Titanic", but no vertical will be shown again in the following pages unless the user types a new query. However, the goal of E-commerce search is quite different from web search [45]. With respect to aggregated search, the main difference between web and E-commerce lies in that the latter needs to present the aggregated search results in each page to meet user search demand, which requires the system to capture the sequential patterns of user behaviors onto aggregated results.

3.2 Vertical Presentation

The second subtask, vertical presentation, or item presentation, is to rank the candidate items selected from verticals. For instance, should the videos of "violin" be positioned above or below the web links in a page? In general, the system presents the most relevant items at the top, which are more likely to be viewed by users [9, 31]. Existing approaches to vertical presentation can be classified into three types: *pointwise*, *pairwise* and *attention-based* approaches.

Pointwise approaches train an independent classifier per vertical to predict the relevance of the corresponding vertical. *Click models* were proposed to predict the user response (i.e. click or skip) to vertical results [16, 38, 39]. FCM [8] considered the rank of the verticals, their visual salience, and the distance between them and the current item. Markov et al. [22] proposed a click model that estimated different probabilities for results above or below the vertical as well. Others directly trained *vertical-specific classifiers* to predict the relevance score of a source to a query, which was used to decide

the position of each vertical [6, 27, 30]. Ponnuswami et al. [28] trained a gradient boosted decision tree to learn relevance scores and ranked each vertical individually using a computed threshold. Although it is simple and intuitive to implement pointwise models, they suffer from the *relevance ranking* issue that the prediction scores from individual classifiers are not directly comparable.

Pairwise approaches learn to predict the *relative preference* between candidate sources to be displayed in a SERP. Arguello et al. [2] trained one binary classifier for user preference per vertical pair, and derived a final ranking from the predicted preferences. The pairwise method solves the *relevance ranking* issue, but in return, it requires a complex ranking principle among verticals and results in a large number of pairwise classifiers.

The recent study [42] adapted an **attention-based** method that applied different weights to different information sources for relevance estimation. It has gained a significant improvement on ranking relevance, however, it requires a large number of data annotations on relevance score to support model training.

4 HIERARCHICAL SEARCH AGGREGATION

We develop a deep hierarchical reinforcement learning algorithm for aggregated search in this paper. The model consists of a high-level source selection policy and a low-level item presentation policy. The entire process works as shown in Fig. 3. When a new search request arrives, the source selector decides which sources to present on the current page. The item presenter then decides the presentation order of the relevant items from selected sources. The user search session continues when the user scrolls to the next page, and ends when he leaves the platform or types a new query. Both policies are trained with the DQN [24] algorithm. The rewards for policy training come from implicit user feedback, instead of any artificial metrics [3, 43], so that it can learn from user behaviors straightforwardly.

In the following subsections, we will present the details of the two policies, deep Q network architecture, and model optimization. The notations used in this paper are summarized in Table 1.

Notation	Description
x	Search request
$\mathcal{SR}_j$	Set of all items of source type j
m_j^k	K-th item of source type j
$n_j(x)$	The number of items of type j given the request x
p, P	Slot position on the SERP, Result presentation
s, $\mathcal{S}$	State, State set
a, $\mathcal{A}$	Primitive action, Primitive action set
o, O	Option, Option set
r^e, r^i	Extrinsic reward, Intrinsic reward
γ	Discount factor
R	Return (discounted cumulative reward)
μ, π	Source selector policy, Item presenter policy

Table 1: Notations of aggregated search and HRL.

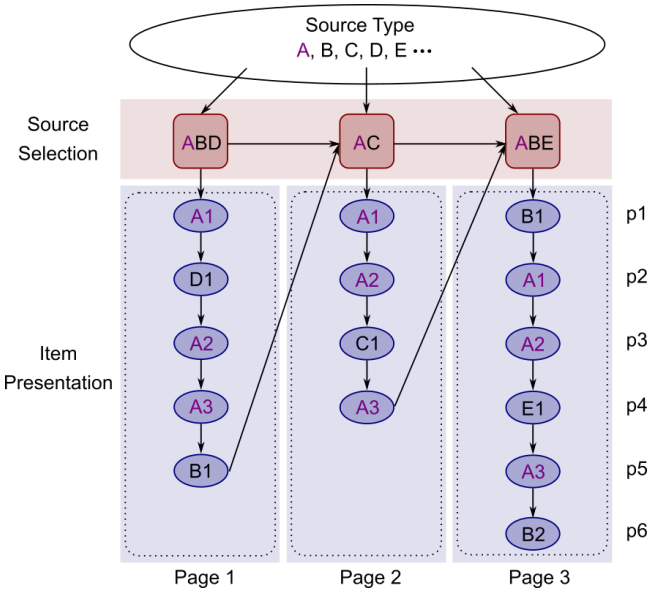

Figure 3: The hierarchical architecture on aggregated search. This example of user search session is composed of three search requests (page 1 to 3). A stands for the products, and $B, C \cdots$ for the items of other source types.

4.1 Source Selection with High-level Policy

The high-level RL process aims to decide which sources should be selected in each page. The high-level policy μ perceives the high-level state s^e and selects an option o that tells what sources to display. The option o will trigger the low-level policy to decide how to present the items from these selected sources. The high-level source selector tries to capture the sequential patterns of user behaviors onto aggregated results in different pages.

4.1.1 Option. An option $o \in O$ refers to a multi-step action [33] that selects the relevant sources to display. The core search, which is the product search in our case, must be selected, while verticals can either be selected or not. So if there are N verticals, the size of the option set is $|O| = 2^N$. The agent samples an option o according to μ, then the control of the agent transfers to the low-level policy. The option lasts until the low-level RL process terminates, then the source selector continues to execute the next option.

4.1.2 State. The state $s_t^e \in \mathcal{S}$ of the high-level RL process at time step t is represented by: 1) the search request x, 2) the triggered search source results $\mathcal{SR}(x)$, and 3) the *latest* option $o_{t'}$ where $t - t'$ is the duration of $o_{t'}$. Then the state vector $\phi(s_t^e)$ is the concatenation of the embedding vector of each part. It should be noted that the vector of source type $\mathcal{SR}_j$ is represented by the mean value of all item vectors of that type, i.e. $\mathbf{e}(\mathcal{SR}_j(x)) = \frac{1}{n_j(x)} \sum_k \mathbf{e}(m_j^k(x))$ where $\mathbf{e}(m)$ is the item vector of m. In particular, the vector of a certain source is represented by zeros if the result of that type is empty. Then the vector of entire source results $\mathbf{e}(\mathcal{SR})$ is derived from the concatenation of $\mathbf{e}(\mathcal{SR}_j)$ from all source types.

4.1.3 Extrinsic reward. The extrinsic reward r^e is given to the source selector to evaluate the global performance. It is derived from the mean value of the intrinsic return from the low-level process, which can be formulated as follows:

$$r_t^e = \frac{1}{l} R_t^i = \frac{1}{l} \sum_{k=0}^{l-1} \gamma^k r_{t+k}^i, \tag{1}$$

where l is the duration of the option o_t. The average return is calculated to prevent the tendency that the source selector greedily chooses all triggered sources to obtain more rewards. The definition of intrinsic reward will be introduced soon later.

4.2 Item Presentation with Low-level Policy

The low-level RL process aims to decide the presentation order of the items from the candidate sources $\overline{\mathcal{SR}}(x)$ chosen by the high-level source selector. Note that the items from each source have been ranked according to its private vertical, and all items from verticals can be interleaved with results from other sources. Each display position is regarded as a slot and the entire presentation subtask starts by filling an item into the top slot of the page, and then the second slot, etc. At each step, the low-level RL policy π takes the corresponding option o and the low-level state s^i as input, and outputs a primitive action a that tells which source to display at the current position. As a result, the agent does not give a relevance score to each item thus avoiding the relevance ranking issue.

Considering that the source selector has notified which sources are selected to display, the number of slots on the page can be determined by $l = \sum_{\mathcal{SR}_j \in \overline{\mathcal{SR}}} n_j(x)$. The agent reaches the subgoal when all the slots in a page have been filled, thus the number of slots is equal to the time duration of the option, and the termination function $\beta : \mathcal{S} \rightarrow \{0, 1\}$ is deterministic in our setting. Once it is terminated, the control of the agent will return to the high-level process.

4.2.1 Primitive action. A primitive action $a \in \mathcal{A}$ refers to a one-step action that chooses a certain source type, so $|\mathcal{A}| = 1 + N$ given the core search and N verticals. Each source can be seen as a stack with its items in descending order of relevance to the search request x. If a certain source is selected according to π, the top item at the stack of this source is displayed in the current position, as shown in Fig. 4. When a source stack is empty, the source is excluded from the action space. In this way, we simplify the relevance ranking problem into a slot filling process to mix different sources: we only need to decide from which source the top item should be selected, instead of evaluating the relevance of each source item. Note that the relative order of the items within the same source type remains unchanged, the agent simply utilizes the existing ranking list.

4.2.2 State. The state $s_t^i \in \mathcal{S}$ of the low-level RL process at time step t is represented by: 1) the search request x, 2) the *top items* of candidate sources $top(\overline{\mathcal{SR}}(x))$, 3) the last primitive action a_{t-1}, and 4) the option o that launches the current subtask, and the state vector $\phi(s_t^i)$ is the concatenation of all vectors of each corresponding part. In order to give an intuitive understanding of *top items*, we take Fig. 4 as an instance. $top(\overline{\mathcal{SR}}) = [A5; B1; C2]$ before the agent presents the item at 6th slot position, then $top(\overline{\mathcal{SR}}) = [A5; B2; C2]$ after it chooses the source type B. So naturally, the vector of candidate sources $\mathbf{e}(top(\overline{\mathcal{SR}}))$ is represented by the concatenation of the top item vector $\mathbf{e}(m_j^{top})$ from each selected source. In addition, similar to [29], we put options into the low-level state representation as an additional input throughout the low-level RL process, to make the selected sources available for the policy π.

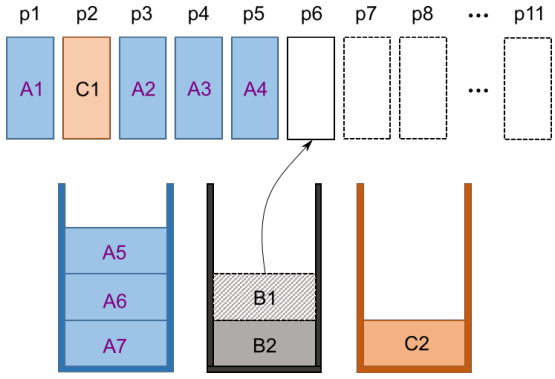

Figure 4: Illustration of the proposed slot filling scheme. There are three sources (A,B,C) and 7+2+2=11 slots in this example.

4.2.3 Intrinsic reward. The intrinsic reward r^i is provided for the item presenter to indicate how well a particular subtask is completed. User clicks and transactions are used to form the intrinsic reward, which is computed as below:

$$r_t^i = \lambda * click + (1 - \lambda) * \min(ln(1 + pay), \delta), \tag{2}$$

where λ is a weight factor, $click \in \{1, -1\}$ indicates whether the user clicks or skips the current slot, and $pay \in [0, +\infty)$ is the price of the product that the user buys. When the user does not buy anything, the second term is zeroed as $pay = 0$. The second term is also clipped by the value δ to reduce the impact of extreme observations. In addition, if there is no click or transaction during the current subtask, a small negative value is added to the intrinsic reward at the last time step as a penalty.

4.3 Q Network Framework

Deep Q-learning uses a multi-layer neural networks with parameters θ to estimate the Q function. Given an n-dimensional state s, it outputs a vector of Q-values over the actions $Q(s, \cdot; \theta) : \mathbb{R}^n \rightarrow \mathbb{R}^{|\mathcal{A}|}$.

In our algorithm, both high-level and low-level policies leverage the same Q network framework, as shown in Fig. 5. We apply DRQN [14] by considering the previous context with a recurrent structure to integrate the underlying correlation between the previous state and current observation. Eq. 3 refers to the hidden layer and RNN layer in Fig. 5.

$$\begin{aligned} \mathbf{w}_t &= \beta(\mathbf{W}_\phi \cdot \phi(s_t) + b_\phi), \\ \mathbf{h}_t &= GRU(\mathbf{w}_t, \mathbf{h}_{t-1}), \end{aligned} \tag{3}$$

where $\phi(s)$ is the vector representation of the state s, β is *Leaky ReLU* activation function.

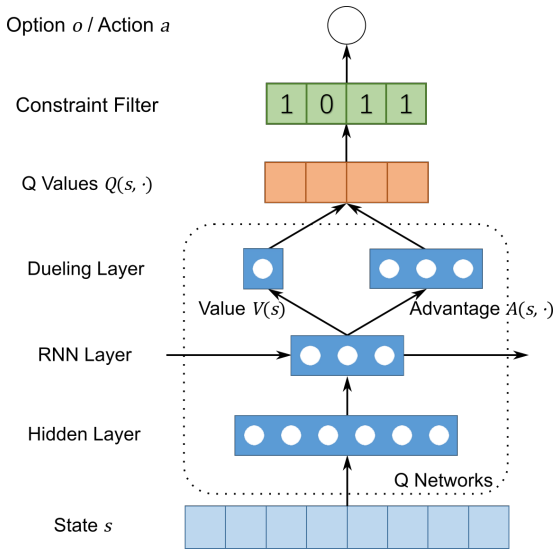

Figure 5: Q network framework used in both high and low level policies.

Besides, the dueling architecture [40] is introduced as well to get a more precise policy evaluation in the presence of many similar-valued actions. It consists of two streams that represent the value and advantage functions, and the two streams are combined afterward to provide a better estimate of Q-values, i.e. the dueling layer of Fig. 5.

$$Q(s,a;\theta) = V(s;\theta) + (A(s,a;\theta) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta)). \quad (4)$$

The *constraint filter* is to explicitly perform an element-wise mask operation on the Q-values to meet some complicated commercial needs. In this way, commercial constraints in a special situation can be easily satisfied in this framework. For instance, not displaying any blog posts in the first page can be dealt with by the source selector via excluding the corresponding source type from the action space. Similarly, displaying a particular source item at a pre-specified position can be handled by the item presenter via excluding the corresponding slot from the slot sequence.

4.4 Hierarchical Policy Learning

Standard DQN is used to minimize the squared error between the target (Bellman optimality condition) $r + \gamma \max_{a'} Q(s', a'; \theta)$ and its estimate $\hat{Q}(s, a; \theta)$. In the context of HRL, the top-level policy μ tries to minimize the following loss function at each time step t:

$$\begin{aligned} L_\mu(\theta_t^e) &= \mathbb{E}_{(s,o,r,s')\sim D^e}[(y_t^e - Q_\mu(s,o;\theta_t^e))^2], \\ y_t^e &= r + \gamma^l Q_\mu(s', \arg\max_{o'} Q_\mu(s',o';\theta_t^e); \theta_t^{e-}), \end{aligned} \quad (5)$$

where l is the number of steps the option o lasts. Here we apply experience buffer [23] to smooth over changes in the data distribution. We also employ a target Q network [24] with parameters θ^- to decrease the possibility of divergence or oscillations during the training process, and double Q-learning [13] to reduce the observed over-estimations caused by traditional Q-learning. Similarly,

Algorithm 1: Deep hierarchical policy learning for aggregated search

Input: Item sets with multiple source types $\mathcal{SR}$ and their search services.

Initialize the two policies with parameters $\{\theta^e, \theta^i\}$
Initialize replay memories $\{D^e, D^i\}$ respectively
$\theta^{e-} = \theta^e$, $\theta^{i-} = \theta^i$
foreach *user search session* **do**
  Initialize high-level state s^e with current search request x
  repeat
    // Page-level source selection
    Set option $o =$
    $\begin{cases} \text{random sample option } o \text{ from } \mathcal{O}, & random() < \epsilon^e \\ \arg\max_{o'} Q^\mu(s^e, o'; \theta^e), & otherwise \end{cases}$
    Initialize low-level state s^i and option duration l according to o
    for $k = 0$ **to** $l - 1$ **do**
      // Slot-level item presentation
      Set primitive action $a =$
      $\begin{cases} \text{random sample action } a \text{ from } \mathcal{A}, & random() < \epsilon^i \\ \arg\max_{a'} Q^\pi(s^i, o, a'; \theta^i), & otherwise \end{cases}$
      Execute action a, observe intrinsic reward r^i (cf. Eq. 2) and next state s'^i
      Store transition (s^i, o, a, r^i, s'^i) in D^i
      Randomly sample mini-batches from D^i
      Perform gradient descent on $L_\pi(\theta^i)$ (cf. Eq. 6 & 8)
      $s^i = s'^i$
      Assign $\theta^{i-} = \theta^i$ every C^i steps
    end
    Obtain extrinsic reward r^e (cf. Eq. 1) and next state s'^e
    Store transition (s^e, o, r^e, s'^e) in D^e
    Randomly sample mini-batches from D^e
    Perform gradient descent on $L_\mu(\theta^e)$ (cf. Eq. 5 & 7)
    $s^e = s'^e$
    Assign $\theta^{e-} = \theta^e$ every C^e steps
  until *the session ends*
end

we learn the low-level policy π by minimizing the following loss function:

$$\begin{aligned} L_\pi(\theta_t^i) &= \mathbb{E}_{(s,o,a,r,s')\sim D^i}[(y_t^i - Q_\pi(s,o,a;\theta_t^i))^2], \\ y_t^i &= r + \gamma Q_\pi(s', o, \arg\max_{a'} Q_\pi(s',o,a';\theta_t^i); \theta_t^{i-}), \end{aligned} \quad (6)$$

given the option o from μ.

Differentiating the loss function with respect to the weights, we arrive at the following gradient for the high-level policy:

$$\nabla_{\theta_t^e} L_\mu = \mathbb{E}_{(s,o,r,s')\sim D^e}[(y_t^e - Q_\mu(s,o;\theta_t^e))\nabla_{\theta_t^e} Q_\mu(s,o;\theta_t^e)], \quad (7)$$

and similarly, the gradient for the low-level policy yields as below:

$$\nabla_{\theta_t^i} L_\pi = \mathbb{E}_{(s,o,a,r,s')\sim D^i}[(y_t^i - Q_\pi(s,o,a;\theta_t^i))\nabla_{\theta_t^i} Q_\pi(s,o,a;\theta_t^i)]. \quad (8)$$

Note that the initial hidden state of RNN layer should be carried forward from its previous values $\mathbf{h}_{t-1}$ when sampling actions, but we zero it at the start of the update so that we do not need to save $\mathbf{h}_{t-1}$ into replay buffers. The convergence and performance of such complexity reduction are guaranteed [14]. In addition, we clip the error term through *Huber Loss* to improve the stability of the algorithm.

The entire training process is described in Algo. 1.

5 EXPERIMENTAL SETUP

To evaluate the performance of our proposed approach, we carried out experiments on the Taobao Search platform, which is one of the largest E-commerce search services in the world, with over 1 billion user clicks every day. All models in this paper adopt the online-learning paradigm and are trained on the hourly real-time search log streaming data.

The online experiment methodology we adopted is called bucket testing, also known as A/B test. In the bucket testing system of Taobao Search, several test buckets are set up. And all users are randomly hashed into these buckets based on their user ids. Each bucket has the same number of users, and the same distribution of users. Then each algorithm is deployed on one bucket. The performance of an algorithm is estimated using the metrics calculated on the bucket it is deployed on. In this paper, our online bucket testing lasted for two weeks, a period long enough to ensure the statistical stability of our test results.

5.1 Features and Parameter Settings

We aggregated two verticals, namely topic and group verticals, into the results of product search. For each RL policy in our model, the components of state s (i.e. search request x, search source results $\mathcal{SR}(x)$, etc.) are represented by a number of features, which mainly contains 1) the query features: including the word-segmented tokens of the query; 2) user features: including user gender, age, items user clicked in the previous page; 3) page features: the current page number; 4) source features obtained from each search services, including the type and title of a topic group or a blog post.

The parameter settings for the high-level source selector and low-level item presenter are provided in Table 2.

Hyper-parameter	Setting	
	Source selector	Item presenter
Learning rate	1e-2	1e-4
Optimization algorithm	RMSProp	RMSProp
Memory D size	5e4	5e5
Mini-batch size	32	32
Target θ^- update period C	1e3	1e4
State feature vector size	48	56
Hidden layer size	28	24
RNN layer size	16	12
Discount factor γ	0.95	
Weight factor λ in Eq. 2	-	0.3
Clipped to δ in Eq. 2	-	3

Table 2: Hyper-parameter settings

5.2 Baselines and Evaluation Metrics

In order to show the effectiveness of our HRL model, we implemented several aggregation methods for comparison, which can be grouped into two categories: *rule-based* and *learning to aggregate* methods, while the latter can be separated into two types: *one-stage* and *two-stage* methods according to the process flow. The *one-stage* method is introduced to verify the hierarchical decomposition in the aggregate search, and the *two-stage* method is carried out to validate the RL formulation in each subtask. The details of our baselines are introduced as follows:

Rule: a rule-based method, where the rule is manually made for search aggregation according to abundant experience in search product design. The rule claims that the first page should not present aggregated results, and the topic and blog verticals will be alternately displayed at fixed positions from the second page when a topic group or a blog post is available from backends. In a page, the rule displays a topic group at the 4th position, and a blog post at the 9th position. Note that this simple rule is the benchmark for comparison with other methods.

Flat RL: a one-stage method. In this paradigm, the aggregation task is simplified into one stage: for each aggregation request, just choose the optimal template from a set of predefined templates. Seven templates are used in this method and are provided in Fig. 6. Each template clarifies exactly what to fill in each slot. These templates are designed by human experts familiar with Taobao Search to ensure good customer experience. The aggregation problem is solved by a single DQN agent. The state representation and the reward of this DQN are the same with the high-level state and the extrinsic reward defined in subsection 4.1 respectively.

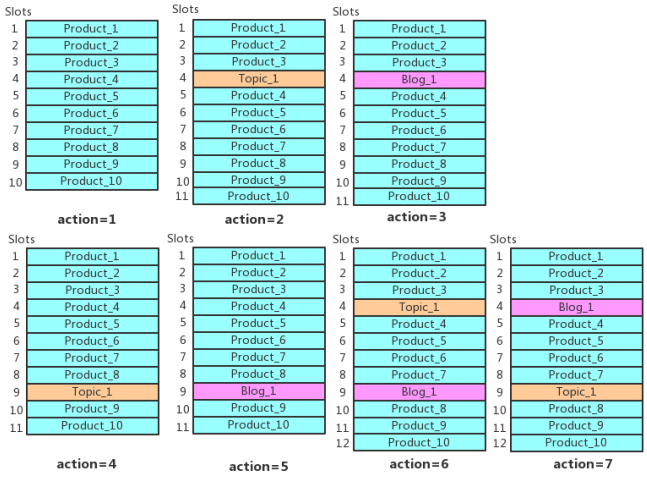

Figure 6: The SERP templates for our 1-stage DQN baseline.

BC+RM: a two-stage method similar to [28]. For *vertical selection*, it uses a binary classifier for each source to decide whether to present this source in a page. The binary classifiers are Neural Networks with 3 hidden layers. The training data for the binary classifiers come from online search log, where each page view is a sample. For a vertical in the page, if it is clicked by a user, then

Method	Topic group						Blog post					
	CTR		ADT		COV		CTR		ADT		COV	
	value (1e-2)	gain	value	gain	value (1e-2)	gain	value (1e-2)	gain	value	gain	value (1e-2)	gain
Rule	5.24	-	10.56	-	5.60	-	3.43	-	75.78	-	6.47	-
Flat RL	6.08	15.99%	10.62	0.54%	5.98	6.79%	3.92	14.29%	76.69	1.20%	6.76	4.48%
BC+RM	5.59	6.76%	10.68	1.09%	6.03	7.68%	3.69	7.58%	76.99	1.59%	6.75	4.33%
BC+RL	5.60	6.77%	10.66	0.92%	6.00	7.32%	3.73	8.75%	76.69	1.21%	6.82	5.41%
RL+RM	5.52	5.29%	10.75	1.75%	6.09	8.75%	3.60	6.12%	76.46	0.91%	6.87	6.18%
HRL	7.34	**40.07**%	10.65	0.89%	5.95	6.25%	4.25	**23.91**%	76.53	0.98%	6.86	6.03%

Table 3: Metrics of online bucket testing on two verticals.

it is a positive sample, otherwise a negative sample. For *vertical presentation*, it uses a 3-hidden-layer Neural Network regression model to score each item. The regression target of the regression model is the same with the intrinsic reward in Eq. 2.

BC+RL: a two-stage method similar to *HRL*, which replaces the high-level source selector with multiple binary classifiers as described in *BC+RM*, but the RL policy for the item presenter remains unchanged.

RL+RM: a two-stage method similar to *HRL*, which replaces the low-level item presenter with a regression model as described in *BC+RM*, but the RL policy for the source selector remains unchanged.

Note that all *learning to aggregate* methods were pre-trained on offline data by *behavioral cloning* prior to online experiments, so as to mitigate the slow learning and poor performance in the early stage of online training. The offline data, which contain 60,000 search sessions covering a broad query topics with an average of 11.3 items per page and 13.4 pages per session, were collected from the user search logs on the Taobao platform.

To evaluate the performance of the algorithms, we use some common evaluation metrics in search aggregation, including:

Click Through Rate (CTR): the ratio of users who click on a specific vertical to the number of total users who view the vertical on a page.

Average Dwell Time (ADT): the mean value of a user's dwell time in seconds on a specific vertical.

Coverage (COV): the ratio of the number of slots occupied by a specific vertical to the total number of slots.

Moreover, we also include an important metric in the E-commerce industry: the *Gross Merchandise Volume* (GMV) of each search service, which is the total revenue of sales made by the product search, or induced by a vertical in SERPs within a fixed time period.

6 RESULTS

6.1 Online Performance

We show the results of online bucket testing in terms of CTR, ADT and COV of different verticals in Table 3. Among these three metrics, the most important one is CTR, which indicates whether users clicked the verticals in the aggregated SERP. For each metric in Table 3, we present its value in the "value" column, and its relative improvement over the *Rule* baseline in the "gain" column. The results in Table 3 show that all the baselines and our HRL model achieve similar improvement rate on ADT and COV. However, it is worth noting that the result on CTR differs remarkably among all the methods.

In general, **HRL** achieves much better CTR improvement than all the baselines on both topic and blog verticals. The comparison to **Flat RL** shows that our hierarchical decomposition of the RL process is very effective, achieving better CTR from the historical user behaviors across different pages as well as the immediate user feedback in the current page. And the result also indicates that it is more sensible to allow a model to choose from any positions when displaying heterogeneous items, rather than restrict the positions where a vertical can be presented in a page. While the *one-stage* method can only use limited aggregation templates to meet the trade-off between the model performance and the exponential number growth of templates due to the number of sources, the hierarchical decomposition of *HRL* overcomes the exponentially large action space problem and allows the agent to explore all possible source combination and item permutation.

The comparisons to *BC+RM*, *BC+RL*, *RL+RM* show that the sequential RL modeling in both high-level and low-level aggregation subtasks is critical for a good performance. A common feature of these discriminant models is that they all follow the vertical selection and presentation pipeline, but make point-wise decisions, without the passing of states between sequential decisions as in *HRL*. **BC+RL** considers the user feedback only based on the current page, while *HRL* tracks the user behaviors from page to page to understand user intents. The result verifies that it is essential to capture sequential patterns of user behavior in aggregated E-commerce search. Similarly, the comparison to **RL+RM** demonstrates that *HRL* also consistently outperforms the baseline that simply ranks items from heterogeneous sources since it suffers from relevance ranking issue by giving scores to each vertical, which concludes that our slot filling process aggregates a more reasonable SERP.

Another important metric, the GMV of the aggregated E-commerce search, that we care about is shown in Table 4. It should be noted that, due to Alibaba's business policy, we temporarily cannot expose the absolute values of GMV. Hence, we report relative GMV increase over *Rule* instead, and this will not affect performance comparison. Because the GMV of aggregated search is the total merchandised volume of all sources, we report not only the GMV of topic and blog verticals, but also the GMV of product search in Table 4.

Method	Topic group	Blog post	Products
Flat RL	38.4%	12.31%	-0.47%
BC+RM	19.32%	5.83%	-0.41%
BC+RL	20.78%	6.00%	-0.37%
RL+RM	17.60%	5.54%	-0.20%
HRL	**56.49%**	**16.93%**	**0.56%**

Table 4: The relative GMV increase over *Rule*. The GMV of a vertical is computed through the sales value guided by the presented items of that vertical.

As shown in Table 4, *HRL* is the only method that increases the GMV of product search. Although other baselines achieve GMV increase on the topic and blog verticals, their GMVs on product search all decrease. For example, *Flat RL* brings a marked GMV improvement on topic and blog verticals, at the expense of the largest GMV drop on product search though. This means that *HRL* provides the best user experience in its aggregated SERP, because even a growth on the COV of topic and blog verticals does not reduce users' purchases in product search. Moreover, *HRL* also achieves the best GMV increase on the topic and blog verticals, leading to the best overall GMV increase among all methods.

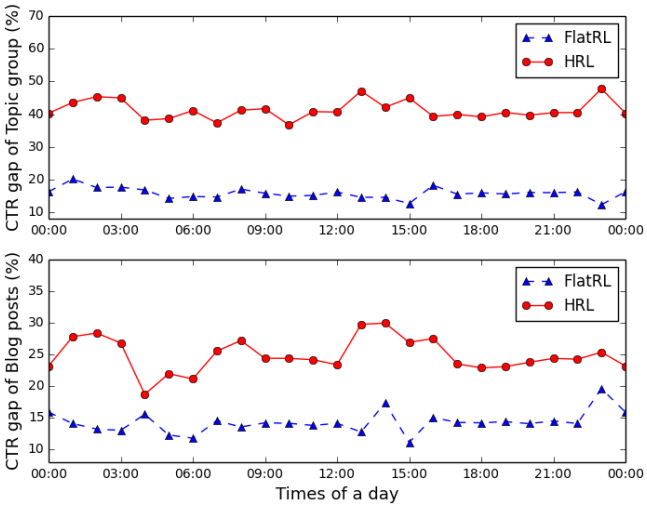

Figure 7: The relative CTR increase over *Rule* in different hours. Each mark indicates the result in the time span between the current time and next full hour.

To provide more insights on the online experiments, we group the two-week statistics by different hours of a day to observe the performance variation over time. For simplicity, we only compare the relative CTR increase of the two most competitive methods: *Flat RL* and *HRL*. Fig. 7 shows the trend of their CTR increase over the baseline *Rule* during different time periods of a day. It can be seen that the CTR increase does not variate too much at different hours of a day, and *HRL* has consistently better CTR improvement than *Flat RL*.

6.2 User Evaluation

In order to study the user experience change caused by our HRL aggregation method, we invite 50 users of Taobao Search to evaluate the quality of our aggregated result against that of the rule-based baseline: *Rule*, which is the old online aggregation method of Taobao Search and has been online serving for several years. The evaluation is performed on a fixed query set: 200 popular queries covering 13 salient product categories in Taobao. This query set is selected by third-party experts on Taobao platform. A user is asked to search each query in both our and the old aggregation environments simultaneously, without knowing which environment is the old one. And the aggregated results of the two environments, designated by A and B, are presented to the user side-by-side. The user can browse the results, click items, scroll to the next page, and so on. And the user provides a grade from {"System A better", "Same/cannot judge", "System B better"} on each vertical for this query. Then we translate the user's grade to {"New System better", "Same/cannot judge", "Old System better"}, where the "New System" refers to *HRL* and the "Old System" refers to *Rule*. The distribution of user grades on each vertical is shown in Table 5.

Vertical	New system better	Same	Old system better
Topic	30.35%	67.98%	1.67%
Blog	26.14%	72.53%	1.33%

Table 5: The distributions of user evaluation grades.

The results in Table 5 show that user experience grades are the same on about 70% of the queries. In spite of this, *HRL* provides better user experience than the baseline most of the times on the rest 30% of the queries.

To understand how *HRL* improves user experience, every time when a user thinks the new system is better, we inquire about his reason for this preference. And we also ask them to group their reasons into three groups as follows:

- Better **behavior relevance**: In *HRL*, the presented verticals are more relevant to the user's previous behaviors.
- Better **query relevance**: In *HRL*, the presented verticals are more relevant to the query.
- **Others**: other reasons.

Vertical	Behavior relevance	Query relevance	Others
Topic	42%	33%	25%
Blog	37%	35%	28%

Table 6: Distributions of reasons why new system is better.

The results in Table 6 show that *HRL* contributes to both user behavior relevance and query relevance. We present two cases here to give some clue that how user experience is improved.

A case for better user behavior relevance is as follows: a user issues the query "running shoes", and clicks several products with the same brand: "Nike" on the first page, then on the next page the topic group titled "Nike shops with discounts" is presented at the

second position in our HRL aggregated result, which is very relevant to the user's behavior. In contrast, no topic group is presented by the baseline in the aggregated result on the next page. In *HRL*, user behaviors in previous pages are encoded into the RNN layer hidden states in the Q network shown in Fig. 5. So user behaviors in previous pages are passed on to the next page for decision making in *HRL*, which helps perceive user behavior relevance. Besides, *HRL* can also capture sequential patterns of user behavior by learning from the long-delayed user feedback across the whole user search session.

A case for better query relevance is as follows: a user issues the query "women's long dress autumn Vero Moda", which is very specific in brand, season and style. The blog post titled "Introduction to the world of women's dresses", which gives a general introduction to women's dresses, is not presented in the aggregated result of *HRL*, whereas it is presented in the baseline's aggregated result. The user prefers our aggregated result because she thinks the blog post is too general thus not meeting her specific needs. With query and blog title features as input, *HRL* is able to identify that the general introduction of the blog post is not relevant enough to the query she inputs.

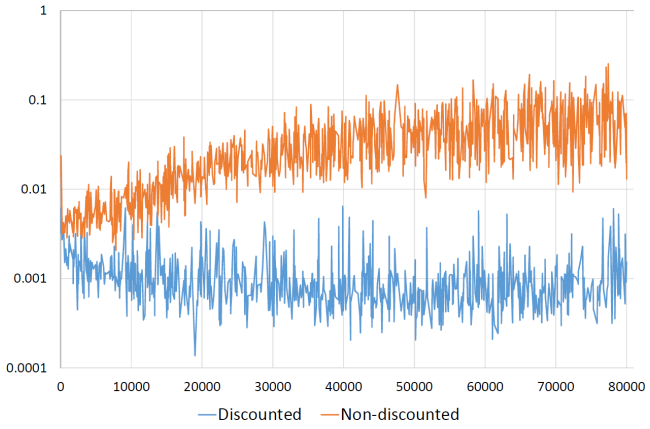

Figure 8: Loss curve of high-level source selector between different training strategies. Blue curve for strategy (I) and orange curve for strategy (II). The horizontal axis is the training iteration, and the vertical axis is the loss value in log scale.

6.3 Training Analysis

To offer a detail training analysis of HRL, we investigate the training procedure of the high-level RL process that takes multi-step actions. Two strategies are designed for the source selector. (I) One is the proposed one in section 4.1 & 4.4 that the extrinsic reward is the mean value of the intrinsic return from the item presenter, i.e. the same as Eq. 1, and the optimization target of Q network (y_t^e in Eq. 5) is discounted by γ^l. (II) The other is to regard the high-level RL process as a simple RL process that takes one-step actions. Under this setting, the extrinsic reward is directly equivalent to the mean value of the intrinsic rewards, i.e. $r_t^e = \frac{1}{l}\sum_{k=0}^{l-1} r_{t+k}^i$, then the training target of source selector policy in Eq. 5 is similar to that in Eq. 6: $y_t^e = r + \gamma Q_\mu(s', \arg\max_{o'} Q_\mu(s', o'; \theta_t^e); \theta_t^{e-})$.

The learning curves of the two strategies are demonstrated in Fig. 8, where the blue and orange curves correspond to strategy (I) and (II) respectively. It is clear that the proposed strategy (I) converges well as the training procedure proceeds while the strategy (II) diverges dramatically. This result shows that it is necessary to handle a multi-step RL process in a proper way to guarantee its convergence. From another perspective, the strategy (I) puts more weight on the items presented in the top slots, which is consistent with the position bias in most search area that top-ranked search results attract more clicks, so the agent can better understand the sequential patterns of user behavior and learn from such data effectively.

7 CONCLUSION AND DISCUSSION

We have studied the search aggregation problem in a large E-commerce search service and proposed an effective HRL aggregation method. Different from web search aggregation, the E-commerce aggregation needs to be performed for each page, leading to multiple and sequential aggregation decisions for one query. In light of this, we propose a hierarchical aggregation model for E-commerce search aggregation. Our HRL model naturally fits the two-stage decomposition of the aggregated search task: a high-level RL for source selection and a low-level RL for item presentation. The source selector models user interactions on aggregated results across pages to capture the sequential patterns of user behavior, and the item presenter formulates a slot filling process that sequentially presents the heterogeneous items to avoid the relevance ranking issue. In this manner, the HRL model can fully learn from user feedback in the current page as well as in the entire search session.

We perform online bucket testing on Taobao Search platform and compare our HRL method with several baselines. The results show that our method contributes a substantial improvement over the baselines for all verticals. Moreover, it also boosts the GMV of product search and achieves the best overall GMV among all the models. User study also demonstrates that it improves user experience by displaying better aggregated results that are relevant to user behaviors and needs. Note that our method can easily combine new verticals into an aggregated search system.

One limitation of the proposed algorithm may lie in the automatic offline evaluation. All we can get from the offline data is the records of user behaviors on the presented results, so it is difficult to infer the user feedback on the unseen SERPs without full annotations of the heterogeneous item relevance as we train our model in a RL setting. However, sufficient experiments including bucket testing and user study indicate that our method indeed shows its effectiveness on the search aggregation. Future research will be focused on the application of our HRL model in different E-commerce verticals and other scenarios.

ACKNOWLEDGEMENTS

This work was jointly supported by the National Key R&D Program of China (Grant No. 2018YFC0830200) and the National Science Foundation of China (Grant No.61876096/61332007). We would also like to thank Mr. Xiaoyi Zeng, our colleague in Alibaba, for constant support and encouragement.

REFERENCES

[1] Jaime Arguello. 2017. Aggregated search. *Foundations and Trends in Information Retrieval* 10, 5 (2017), 365–502.

[2] Jaime Arguello, Fernando Diaz, and Jamie Callan. 2011. Learning to aggregate vertical results into web search results. In *Proc. 20th ACM Int. Conf. Information and Knowledge Management*. 201–210.

[3] Jaime Arguello, Fernando Diaz, Jamie Callan, and Ben Carterette. 2011. A methodology for evaluating aggregated search results. In *Proc. 33rd European Conf. Information Retrieval*. 141–152.

[4] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. 2009. Sources of evidence for vertical selection. In *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 315–322.

[5] Andrew G Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13 (2003), 341–379.

[6] Horatiu Bota, Ke Zhou, Joemon M Jose, and Mounia Lalmas. 2014. Composite retrieval of heterogeneous web search. In *Proc. 23rd Int. Conf. World Wide Web*. 119–130.

[7] Marc Bron, Jasmijn Van Gorp, Frank Nack, Lotte Belice Baltussen, and Maarten de Rijke. 2013. Aggregated search interface preferences in multi-session search tasks. In *Proc. 36th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 123–132.

[8] Danqi Chen, Weizhu Chen, Haixun Wang, Zheng Chen, and Qiang Yang. 2012. Beyond ten blue links: Enabling user click modeling in federated web search. In *Proc. 5th ACM Int. Conf. Web Search and Data Mining*. 463–472.

[9] Ye Chen, Yiqun Liu, Ke Zhou, Meng Wang, Min Zhang, and Shaoping Ma. 2015. Does vertical bring more satisfaction?: Predicting search satisfaction in a heterogeneous environment. In *Proc. 24th ACM Int. Conf. Information and Knowledge Management*. 1581–1590.

[10] Aleksandr Chuklin, Anne Schuth, Katja Hofmann, Pavel Serdyukov, and Maarten De Rijke. 2013. Evaluating aggregated search using interleaving. In *Proc. 22nd ACM Int. Conf. Information and Knowledge Management*. 669–678.

[11] Fernando Diaz. 2009. Integration of news content into web results. In *Proc. 2nd ACM Int. Conf. Web Search and Data Mining*. 182–191.

[12] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *Proc. 27th Int. Conf. World Wide Web*. 1939–1948.

[13] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-Learning. In *Proc. 30th AAAI Conf. Artificial Intelligence*. 2094–2100.

[14] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent Q-Learning for partially observable MDPs. In *Proc. 29th AAAI Conf. Artificial Intelligence, Fall Symp. Series, Sequential Decision Making for Intelligent Agents*. 29–37.

[15] Dzung Hong, Luo Si, Paul Bracke, Michael Witt, and Tim Juchcinski. 2010. A joint probabilistic classification model for resource selection. In *Proc. 33rd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 98–105.

[16] Luo Jie, Sudarshan Lamkhede, Rochit Sapra, Evans Hsu, Helen Song, and Yi Chang. 2013. A unified search federation system based on online user feedback. In *Proc. 19th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 1195–1203.

[17] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. 2014. Aggregated search: A new information retrieval paradigm. *Comput. Surveys* 46, 3 (2014), 41.

[18] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proc. 30th Annu. Conf. Neural Information Processing Systems*. 3675–3683.

[19] Or Levi, Ido Guy, Fiana Raiber, and Oren Kurland. 2018. Selective cluster presentation on the search results page. *ACM Transactions on Information Systems* 36, 3 (2018), 28.

[20] Zeyang Liu, Yiqun Liu, Ke Zhou, Min Zhang, and Shaoping Ma. 2015. Influence of vertical result in web search examination. In *Proc. 38th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 193–202.

[21] Bo Long and Yi Chang. 2014. *Relevance ranking for vertical search engines*. Morgan Kaufmann Publishers Inc.

[22] Ilya Markov, Eugene Kharitonov, Vadim Nikulin, Pavel Serdyukov, Maarten De Rijke, and Fabio Crestani. 2014. Vertical-aware click model-based effectiveness metrics. In *Proc. 23rd ACM Int. Conf. Information and Knowledge Management*. 1867–1870.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *Proc. 27th Annu. Conf. Neural Information Processing Systems, Deep Learning Workshop*.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[25] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for relevance and display preferences in complex presentation layouts. In *Proc. 41st Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 845–854.

[26] Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning. In *Proc. 22nd Conf. Empirical Methods in Natural Language Processing*. 2231–2240.

[27] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Desmond Brand, and Tapas Kanungo. 2011. Model characterization curves for federated search using click-logs: predicting user engagement metrics for the span of feasible operating points. In *Proc. 20th Int. Conf. World Wide Web*. 67–76.

[28] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Qiang Wu, Ran Gilad-Bachrach, and Tapas Kanungo. 2011. On composition of a federated web search result page: Using online users to provide pairwise preference for heterogeneous verticals. In *Proc. 4th ACM Int. Conf. Web Search and Data Mining*. 715–724.

[29] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *Proc. 32nd Int. Conf. Machine Learning*. 1312–1320.

[30] Luo Si and Jamie Callan. 2003. Relevant document distribution estimation method for resource selection. In *Proc. 26th Int. ACM SIGIR Conf. Research and Development in Informaion Retrieval*. 298–305.

[31] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and Robert Villa. 2010. Factors affecting click-through behavior in aggregated search interfaces. In *Proc. 19th ACM Int. Conf. Information and Knowledge Management*. 519–528.

[32] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press.

[33] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112 (1999), 181–211.

[34] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A Hierarchical Framework for Relation Extraction with Reinforcement Learning. In *Proc. 33rd AAAI Conf. Artificial Intelligence*.

[35] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in Minecraft. In *Proc. 31st AAAI Conf. Artificial Intelligence*. 1553–1561.

[36] Gilad Tsur, Yuval Pinter, Idan Szpektor, and David Carmel. 2016. Identifying web queries with question intent. In *Proc. 25th Int. Conf. World Wide Web*. 783–793.

[37] Lauren Turpin, Diane Kelly, and Jaime Arguello. 2016. To blend or not to blend?: Perceptual speed, visual memory and aggregated search. In *Proc. 39th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 1021–1024.

[38] Chao Wang, Yiqun Liu, Min Zhang, Shaoping Ma, Meihong Zheng, Jing Qian, and Kuo Zhang. 2013. Incorporating vertical results into search click models. In *Proc. 36th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 503–512.

[39] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. 2016. Beyond ranking: Optimizing whole-page presentation. In *Proc. 9th ACM Int. Conf. Web Search and Data Mining*. 103–112.

[40] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proc. 33rd Int. Conf. Machine Learning*. 1995–2003.

[41] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov decision process for search result diversification. In *Proc. 40th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 535–544.

[42] Junqi Zhang, Yiqun Liu, Shaoping Ma, and Qi Tian. 2018. Relevance Estimation with Multiple Information Sources on Search Engine Result Pages. In *Proc. 27th ACM Int. Conf. Information and Knowledge Management*. 627–636.

[43] Ke Zhou, Ronan Cummins, Mounia Lalmas, and Joemon M Jose. 2012. Evaluating aggregated search pages. In *Proc. 35th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 115–124.

[44] Ke Zhou, Ronan Cummins, Mounia Lalmas, and Joemon M Jose. 2013. Which vertical search engines are relevant?. In *Proc. 22nd Int. Conf. World Wide Web*. 1557–1568.

[45] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search. In *Proc. 27th Int. Joint Conf. Artificial Intelligence*. 3725–3731.