Hierarchical Reinforcement Learning for Aggregated Search

ABSTRACT

Aggregated search aims at integrating multi-sources results into one Search Engine Result Page(SERP) containing a lot of heterogeneous information. In traditional methods there are three subtasks for the aggregated search: vertical selection, item selection, result presentation. Independent models are trained to solve these problems. In this paper, we propose an end-to-end model to jointly optimize these three subtasks using hierarchical reinforcement learning (HRL), where the interaction between different subtasks is fully considered. The whole search process is divided into two-level RL task. The high-level task for vertical selection, result presentation and the low-level task for item selection. To strengthen the communication between the two agents, a self-supervised learning based state representation method is used to connect the RL agents and achieves a more effective joint learning. Experimental results show that our model obtains significant improvements in search performance metrics.

CCS CONCEPTS

• **Information systems** → **Combination, fusion and federated search**.

KEYWORDS

aggregated search, search ranking, hierarchical reinforcement learning

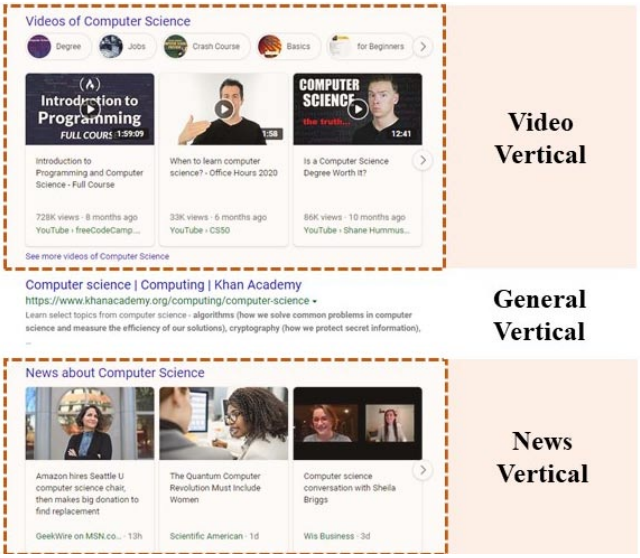

Figure 1: a SERP example for the query "search engine". The items from different verticals including encyclopedia, video, news are aggregated with general blue-link items in this SERP.

In recent years, search engines have changed from only providing blue-links and texts to offering more diversified, professional and complete search results, in order to allow users to obtain richer information in one search. The demands of searching in a specialized field is met by vertical search systems such as Youtube for video, arXiv for academic papers. And the aggregated search systems are to aggregate heterogeneous information from different verticals to construct search result pages(SERP). Figure 1 shows a SERP for "search engine" from a modern search engine, including encyclopedia vertical, video vertical and news vertical to satisfy different needs.

Aggregated search task are typically divided into three sequential subtasks: *vertical selection*, *item selection* and *result presentation*. Vertical selection decides which specific areas of content from the feasible verticals to be presented. Item selection is to select the most relevant items in the chosen verticals to form blocks containing homogeneous content. result presentation is to present the composed blocks on the page in a reasonable way. In the traditional approach, the above subtasks are solved separately using independent models in a pipeline. Although much effort has been spent on improving the performance of individual subtasks, the correlation between subtasks has not been fully considered. Not only does the selection of the vertical affects the item selection and result presentation, selected items also have an impact on the selection of the subsequent verticals and items. In Figure 1, for example, if there are a lot of news-related videos in the video vertical, it might be a better choice to choose a vertical that contains different information than the news vertical.

To address the above issue, we formalize the aggregated search problem as a hierarchical Markov Decision Processes (MDPs). The MDP-based approach, which has been widely used in the field of information retrieval [19, 21] in recent years, constructs the ranking as a decomposable process. At each time step, the model selects a document for the current position, which makes it possible to consider the impact of each decision step on all subsequent decisions. Hierarchical MDPs, on the other hand, decomposes a complex, compound task into MDPs of different granularity. Coarse-grained, high-level MDPs are responsible for overall tasks, while fine-grained, low-level MDPs are responsible for detailed problems. The high-level MDPs guide the low-level MDPs to make decisions and receive feedback from the low-level MDPs at the same time. The MDPs at different levels influence each other and jointly optimize to obtain the optimal solution of the whole problem.

When constructing Hierarchical MDP for aggregated search framework, the above three subtasks can be decomposed into two level MDPs. The high level MDP, called **vertical selector**, decides

DRL4IR '21, July 15, 2021, Virtual Event

which vertical item should be placed in each block, and is in fact responsible for vertical selection and result presentation. The low level MDP, called **item selector**, selects the items from the chosen vertical and sorts them into a block, and is actually in charge of item selection. Policy gradient Hierarchical Reinforcement Learning (HRL) with option framework [16] is used to solve the problem. To smooth the communication between the two components, we additionally devise a special way of state representation for the item selector. Long Short-Term Memory (LSTM) [8] is used to bring in information about the items selected by the item selector. However, training the HRL and LSTM simultaneously using the reward signal alone may lead to an inadequate training of the LSTM. To address this issue, an additional signal is provided using self-supervised learning for joint optimization of HRL and state representations.

In summary, this paper makes the following contributions:

- We model aggregated search framework in a novel hierarchical end-to-end format. The high level for vertical selection and result presentation, while the low level for item selection.
- We introduce hierarchical reinforcement learning to solve this problem. In addition, self-supervised learning based state representation methods are used to strengthen the association of different subtasks.

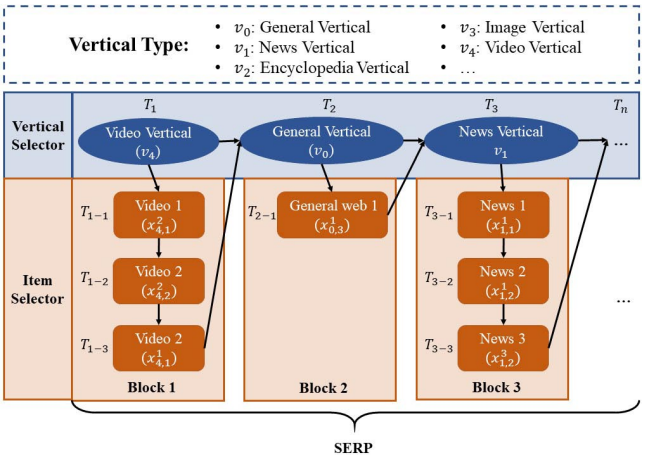

Figure 2: The hierarchical architecture for aggregated search. The high-level vertical selector is responsible for selecting verticals sequentially. Once a vertical is selected, the control is handed to item selector, whose task is to select the appropriate items from the candidate set belonging to selected vertical and form a block.

1 RELATED WORK

1.1 Aggregated Search

Aggregated search can be defined as the integration of search results from various vertical search engines to form a search result page containing various heterogeneous information. Many studies have been done focusing on its subtasks: vertical selection, item selection, result presentation.

Vertical Selection is often seen as a classification problem [9, 13], and a binary classifier is used to decide whether the vertical should be displayed on the SERP. Different types of features are used to express different characteristics of vertical. Both the strongly supervised signal of the ground truth of human judges [3]and the weakly supervised signal obtained based on user behavior [12] can be used for the training of the classifier.

Both **Item Selection** and **Result Presentation** can be considered as the problem that rank candidate sets. Therefore a variety of learning to rank methods can be used to solve these two subtasks. Both pointwise [2, 12] and pairwise [2] ranking functions are used to determine the selection of the items and the layout of the pages. In further, the recent study adapted an MDP-based method [10] that focuses on the association of context information with result presentation. This method draws on the previous modeling approach of using reinforcement learning for ranking, adding context attention and representation learning of context information for optimization.

1.2 Reinforcement Learning

Reinforcement learning (RL) [15] is a way for an agent to learn by "trial and error". The rewards gained from interacting with the environment guide the agent's behavior and enable the agent to accomplish a specific goal. Today, reinforcement learning is widely used, including in the field of information retrieval. [19] and [21] model the ranking problem as a sequential decision problem and use reinforcement learning to improve the relevance and diversity of the ranking, respectively. In E-commerce, multi-scene search [7] and session-based aggregated search [18] are separately solved with multi-agent RL and HRL. Among them, session-based aggregated search has similarities with our scenario. It also aims to generate SERPs containing heterogeneous information, but the difference is that instead of generating only one page for each query, the model has to perform continuous page generation in a session according to the user's click behaviors.

Recently, HRL has also gained a lot of popularity, especially the option framework. It has been successful in many application scenarios by decomposing complex tasks with predefined subtasks of various granularities. In addition to the information retrieval mentioned above, HRL is also useful in relationship extraction [17], MOOC course recommendation [23], and task-oriented dialogues [14]. However, to our best knowledge, we are the first time to use HRL for aggregated search in web search, focusing on integrating the traditional pipeline paradigm into a unified end-to-end model.

2 MAIN APPROACH

We propose a deep hierarchical reinforcement learning model for aggregated search. This model has three components: a high-level RL for **vertical selector**, a low-level RL for **item selector**, and a self-supervised state representation module that connects the two RLs. As shown in Figure 2, vertical selector is responsible for selecting the verticals sequentially, while item selector is responsible for selecting the appropriate items in order from the verticals selected by vertical selector. In this section, we will first introduce the problem definition, followed by the description of the three parts separately.

2.1 Preliminaries

2.1.1 Aggregated Search. We consider the following setting: The goal of aggregated search is to construct a search result page (SERP) of specified length containing results from different verticals for query q. Verticals are predefined and denoted as $V = \{v_0, v_1, ..., v_J\}$ including the general vertical v_0 that contains only traditional blue-link items, also called general web items, as well as other verticals $v_1, v_2, ..., v_J$ containing a variety of heterogeneous information. The items in vertical v_j is denoted as $X_j = \{x_j^1, x_j^2, ..., x_j^I\}$. Each vertical v_j corresponds to multiple search engines, called resource, and denoted as $R_j = \{r_{j,1}, r_{j,2}, ..., r_{j,K}\}$. Given query q, each resource $r_{j,k}$ returns the items $X_{j,k} = \{x_{j,k}^1, x_{j,k}^2, ..., x_{j,k}^I\}$ for query q, which constitutes the candidate set X.

Based on the reality, we have made the following assumptions about the results page:

- An aggregated search page P consists of a set of blocks $B = \{b_1, b_2, ..., b_N\}$.
- Each block consists of one general web item $x_{0,k}^i$ or other vertical items no more than m.
- The items that make up each block b_n can only come from one vertical.
- Except for the general web block, the same type of block can only appear once per page.

Take Figure 2 as an example. The first three blocks of the SERP are video block, general web block and news block. Video block and news block are composed of three items from different resources, while general web block only includes one general web item.

2.1.2 Options Framework. Options framework, an important branch of HRL, is an extension to solve the usual MDPs. Usual MDPs can be defined as $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r >$. $\mathcal{S}$ represents the state space covering all possible states of the agent. $\mathcal{A}$ denotes the action space, which contains all the actions the agent can take. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times S \rightarrow [0, 1]$ denotes the state transition function, which reflects the process of generating the next state s_{t+1} after an action a_t is taken by the agent at a state s_t. $r : \mathcal{S} \times \mathcal{A} \times S \rightarrow \mathbb{R}$ is the reward function, similar to $\mathcal{P}$, except that it returns a reward for current state-action pair. Policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ decides which action a_t is taken in the state s_t. The goal of RL is to find an optimal policy π^* to maximizes the expected cumulative reward, also known as return $R_t = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$, where $\gamma \in [0, 1]$ is a discount factor.

The options framework takes into account the action space with different time granularity in addition to the usual MDP. At each time step, the agent can either choose a primitive action or an option. An option $o \in \mathcal{O}$ is a triple $(\mathcal{I}_o, \mu_o, \beta_o)$, in which $\mathcal{I}_o$ is an initation set, μ_o is an intra-option policy, $\beta_o : \mathcal{S} \rightarrow [0, 1]$ is a termination function. When the agent select an option o_m, a sub-MDP is launched. Sub-MDP starts with the state $s \in \mathcal{I}_{o_m}$ as the initial state. Primitive actions $a_{o_t} \in \mathcal{A}_{o_t}$ are sampled from policy μ_{o_m} to form a sub-trajectory τ_{o_t}, until the termination function β_{o_m} terminates the sub-MDP. Only when the sub-MDP terminats, the new option o_{m+1} can be selected.

2.2 Vertical Selector With High-level RL

The goal of the vertical selector is to sequentially determine which verticals should be displayed on the result page. At each time step t, the high-level RL agent needs to select an option o_t based on policy π to decide which vertical should be placed for the current position. Once an option o_t is selected, a subtask is launched for the low level RL to decide which items should be included in the block for the selected vertical and how they would be arranged.

Option: $\mathcal{O}(s_T^h)$ is all the options available to the agent in state s_T^h. Since the blocks of the same kind cannot be repeated in a page, the size of option set $|\mathcal{O}|$ is decreasing, and if there are N predefined verticals, $|\mathcal{O}| \leq N$. $o_T \in \mathcal{O}(s_T^h)$ selects the appropriate vertical for the position t. The vertical selected by o_T is denoted as $v(o_T)$. When the high level RL agent selects the option, the control is handed over to the low level RL agent. As the low level MDP ends, the control is returned and the selection of the next option is proceeded.

State: The high-level RL state is denoted as $s_T^h \in S$, which can be formulated as a triple $s_T^h = \{q, \mathcal{O}_T, \mathcal{D}_T\}$. q is the query; $\mathcal{O}_t = \{o_0, o_1, ..., o_T\}$ is the options selected so far; $\mathcal{D}_T = \{X_{v(o_0)}, X_{v(o_1)}, ..., X_{v(o_T)}\}$ denotes the global partial ranking list so far, where the items are selected by the low-level agent.To keep the state dimension constant, LSTM is used for state representation. Its design and training method will be explained in section 3.4. The initial state $s_0^h = \{q, \emptyset, \emptyset\}$. The process terminated when the ranking list $\mathcal{D}_T$ reaches the target length.

Extrinsic reward: The extrinsic reward r^h is given to reflect how relevant the vertical chosen by vertical selector is to query q, how well the vertical is sorted, and how well the item is sorted in the whole page. Therefore, we use the F1 of vertical selection, the cumulative gain-based metric NDCG (normalized Discounted Cumulative Gain) [11] and NDCG-IA metrics [1] as the reward signal, where NDCG-IA is the derivative of NDCG, considering both the diversity and relevance of the page content. To avoid sparse rewards, we use the increment of these two metrics during the option o_t as the reward signal, which can be formulated as:

$$\begin{aligned} r_T^h = & \alpha \times (\mathrm{F1}_T - \mathrm{F1}_{T-1}) \\ & + \beta \times (\text{NDCG-IA}_T^h - \text{NDCG-IA}_{T-1}^h) \\ & + (1 - \alpha - \beta) \times (\mathrm{NDCG}_T^h - \mathrm{NDCG}_{T-1}^h) \end{aligned} \tag{1}$$

where α and β illustrate the importance of different metrics.

REINFORCE, a typical policy gradient RL method, is used in vertical selector. The update of the policy network can be calculated as:

$$\theta \leftarrow \theta + \alpha^h \nabla_\theta \log \pi_\theta(o_T | s_T^h) G_T^h \tag{2}$$

where π_θ denotes the vertical selection policy parameterized with θ, G_t denotes the cumulative return of the episode, and α^h denotes the learning rate of the high-level RL policy network.

2.3 Item Selector With Low-level RL

The goal of the item selector is to decide which items should be chosen in the block and how should they be presented. Inspired by the success of MDP-based models in ranking tasks, we formalized the item selection problem as a MDP in a similar with these model.Each time step corresponds to a ranking position, and a primitive action is taken for each time step to select a item for current position.

Primitive action: A primitive action $a_{T,t} \in \mathcal{A}$ selects a item x from a candidate set. The candidate set includes the search results for query q returned by the resources $r_v(o_T)$ corresponding to the selected vertical $v_{(o_T)}$. The index of the item selected by $a_{T,t}$ is denoted as $I(a_{T,t})$

State: The low-level RL state is denoted as $s^l_{T,t} \in S$, which can be formulated as $s^l_{T,t} = \{q, o_T, \mathcal{Z}^t_{v_{(o_T)}}, \mathcal{X}^t_{v_{(o_T)}}\}$. q is the query; o_T denotes the option that launches the current subtask; $\mathcal{Z}^t_{v_{(o_T)}}$ is the partial ranking list for the block so far; and $\mathcal{X}^t_{v_{(o_T)}}$ is the candidate set. The initial state is $s^l_{T,t} = \{q, o_T, \emptyset, X\}$. And the transition function is as follows:

$$\begin{aligned} s^l_{T,t+1} &= \mathcal{P}(s^l_{T,t}, a_{T,t}) \\ &= [q, o_T, \mathcal{Z}^t_{v_{(o_T)}} \oplus I(a_{T,t}), \mathcal{X}^t_{v_{(o_T)}} \backslash I(a_{T,t})] \end{aligned} \tag{3}$$

The selected item is appended to the ranking list and removed from the candidate set. The process terminates when either the global ranking list $\mathcal{D}_T$ reach the target length, or the ranking list $\mathcal{Z}^t_{v_{(o_T)}}$ in the block reach the length limit of the block, or the candidate set $\mathcal{X}^t_{v_{(o_T)}}$ is empty.

Intrinsic reward: Unlike the high-level agent that focuses on the global picture, the low-level agent only focuses on the ranking of the current block. Therefore, NDCG of the current block is used to measure the quality of the ranking.To avoid sparse reward, at each time step, the increment of NDCG is given to the low-level agent as the reward signal, which can be formulated as:

$$r^l_{T,t} = \text{NDCG}^l_{T,t} - \text{NDCG}^l_{T,t-1} \tag{4}$$

Similar to high-level RL, REINFORCE also used to solve the item selection problem. The update of the policy network can be written as:

$$\phi \leftarrow \phi + \alpha^l \nabla_\phi \log \mu_\phi(a_{T,t}|s^l_{T,t}) G^l_{T,t} \tag{5}$$

where μ_ϕ denotes the item selection policy parameterized with ϕ, $G_{T,t}$ denotes the cumulative return of the episode, and α^l denotes the learning rate of the low-level RL policy network.

2.4 Self-supervised State Representation Learning

As we mentioned in INTRODUCTION, our approach is mainly designed to solve the problem of conventional pipeline structure where models are trained separately for different phases, without considering the correlation between them. Therefore, how to connect the high-level and the low-level RL agents is also the focus of our work. It is expected that while the vertical selector can guide the item selector, the items selected by low-level RL agent can also have an impact on the subsequent vertical selection. Thus, a state representation module is designed to deliver low-level information to high-level agent.

Figure 3 shows how the agents communicate with each other. When an option o_T is selected at T, the control is handed over to the item selector. At each time step t for low-level MDP, the agent sample a privmitive action according to its policy and decides which item would be selected for the position t. The embedding of the selected item concatenated with query and the previous

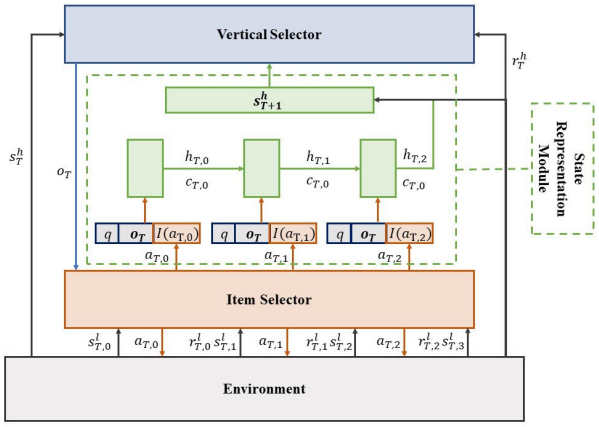

Figure 3: Illustration of how the agents communicate with each other and how they communicate with environment. This figure shows the interaction process within an option

selected option is input to the LSTM model and aggregated with the previously selected item. When the low-level terminates at $t+k$, the hidden state $h_{T,t+k}$ becomes the next state s^h_{T+1} of the high-level RL agent.

The LSTM is formulated as:

$$\begin{aligned} i_{T,t} &= \sigma(W^{(i)} x_{T,t} + U^{(i)} h_{T,t-1}) \\ f_{T,t} &= \sigma(W^{(f)} x_{T,t} + U^{(f)} h_{T,t-1}) \\ e_{T,t} &= \sigma(W^{(e)} x_{T,t} + U^{(e)} h_{T,t-1}) \\ \widetilde{c}_{T,t} &= \tanh(W^{(c)} x_{T,t} + U^{(c)} h_{T,t-1}) \\ c_{T,t} &= f_{T,t} \odot c_{T,t-1} + i_{T,t} \odot \widetilde{c}_{T,t} \\ h_{T,t} &= e_{T,t} \odot \tanh(c_{T,t}) \end{aligned} \tag{6}$$

where $x_{T,t} = [q, o_T, I(a_{T,t})]$.$i_{T,t}$,$f_{T,t}$,$e_{T,t}$,$c_{T,t}$, $h_{T,t}$ denote input gate, forget gate, exposure gate, cell state, hidden state. W, U are learnable parameters. When $t = 0$, the last cell state $c_{T,t-1}$ and the last hidden state $h_{T,t-1}$ come from the previous option.

$$\begin{aligned} c_{T,t-1} &= c_{T-1,t+k} \\ h_{T,t-1} &= h_{T-1,t+k} \end{aligned} \tag{7}$$

where k is the total time step in the previous option. Given that the LSTM may not be adequately trained using only the reward signal, we designed the self-supervised learning method similar to auto-encoder as a supplement.

As shown in 4, the state representation module is considered as an encoder, while an additional LSTM network is built as a decoder to reconstruct the input of the encoder as accurately as possible. The loss function can be formulated as:

$$L(\delta_{en}, \delta_{de}) = \sum_{k=0}^{K} (x_{T,k} - h^{de}_k)^2 \tag{8}$$

where δ_{en} and δ_{de} denote the network parameters of encoder and decoder respectively, K is the length of the sequence, $x_{T,k}$ is the kth input of the enoder, and h^{de}_k denotes the kth hidden state of the decoder.

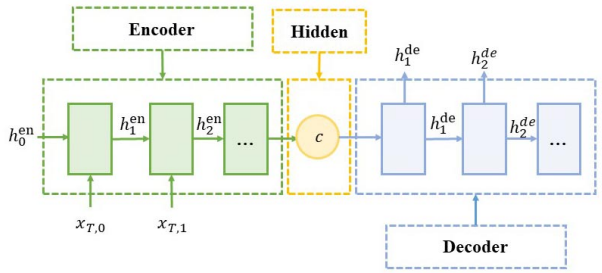

Figure 4: The auto-encoder structure for training the state representation module

Since the state representation module is closely related to High-level RL policy network, the two need to be trained jointly. Therefore, two training methods are proposed, the first one is alternative training and the second one is training with a hybrid loss function. For the former, the RL policy network and the auto-encoder are alternately trained and updated with their respective loss functions. For the latter, these networks are concurrently updated using a unified loss function with two components.The loss function can be formulated as:

$$L(\theta, \delta_{en}, \delta_{de}) = \log \pi_\theta(o_T|s_T^h)G_T^h + \beta_{rep}\sum_{k=0}^{K}(x_{T,k} - h_k^{de})^2 \quad (9)$$

where β_{rep} denotes the weight of loss of representation network.

These two methods offer their own advantages. The alternative training approach avoids adjustment of parameter β_{rep} and is more robust, while training with hybrid loss function introduces the extrinsic reward from the environment as an additional supervisory signal, which is beneficial for the state representation network training. An analysis of the advantages and disadvantages of these two training methods can be found in the experimental section.

3 EXPERIMENTS

3.1 Experiment Setting

3.1.1 Dataset. The experiments are conducted on two public dataset: FedWeb13 and FedWeb14 [6] derived from the TREC federated web search track in 2013 and 2014. Each year, 50 queries are sent to over 150 sub-engines, called resources, belonging to 24 verticals. The example verticals and sub-engines are shown in Table 1 .The snippets and web-pages of up to 10 returned results from the sub-engines are gathered. Five-graded relevance score are given by human judges for every returned results.

3.1.2 Features and parameter Settings. To ensure the fairness of the experiments, we use the gensim's doc2vec model to encode the queries and title of the items as their features. And the features of verticals are obtained by calculating the mean value of the item features belonging to the vertical. The doc2vec model is pre-trained with all the textual information of the dataset.

In order to reduce the dimensionality of the state of the low-level RL and save computation time, we do not include all the returned results from sub-engines when constructing the candidate set for item selector, but use the top 20 items of the returned list.

Vertical	Resources	# Resources
Academic	arXiv.org, CCSB	17
Video	YouTube, Comedy Central	11
Photo/Pictures	Flickr, Getty Images	11
Health	Health Finder, HealthCentral	11
Shopping	Amazon, eBay	10
News	BBC, CNN	10
...	...	...
Books	Goodreads, Google Books	2
Local	Foursquare	1

Table 1: Examples of verticals and resources

Hyper-parameter	Vertical Selector	# Item Selector
Learning rate	1e-5	1e-5
Optimization algorithm	Adam	Adam
Policy network layer size	2	3
Hidden size	128	128
Discount factor	0.99	0.99
Doc2vec feature size	100	100
Weight factor α in Eq.1	0.5	-
Length of the SERP	20	-
Length of the block	-	3
# Candidate items	-	20

Table 2: Hyper-parameter Setting

The policy gradient method: REINFORCE [20] is used for both the high and low level agent, and MLP is used as the policy network. The parameter settings for the vertical selector and the item selector are providing in Table 2.

As for the network ϕ in self-supervised state representation learning, we used a two layer MLP with activate function. The self-supervised learning method and the HRL trained alternately, with training parts exchanged every 1000 episodes.

For each experiment, we perform 5-fold cross validation. For all methods that include RL, the number of episodes trained by the RL agent is the same.

3.1.3 Baselines and Evaluation Metrics. To validate the effectiveness of our proposed approach, we implemented several three-stage methods for comparison. The details of the baselines are introduced as follows:

BC + ISLTR + RPLTR:. a traditional pipeline for aggregated search framework. For vertical selection, a two layer MLP is used as the binary classifer to decide which vertical should be presented in the page. For item selection and result presentation, it uses learning-to-rank (LTR) method. We selected two representative algorithms: RankNet [4] and LambdaRank [5], which are pair-wise LTR and list-wise LTR respectively and are implemented with pytorch [22]. The baseline with RankNet is denoted as **BC + ISRankNet + RPRankNet**, while the baseline with LambdaRank is denoted as **BC + ISLambdaRank + RPLambdaRank**

Method	NDCG@10	NDCG@20	NDCG-IA@10	NDCG-IA@20
BC + ISRankNet + RPRankNet	26.47	26.18	5.78	6.76
BC + ISLambdaRank + RPLambdaRank	**29.16**	**27.39**	**6.48**	7.07
High-level RL + ISRankNet	20.14	24.72	4.75	6.84
High-level RL + ISLambdaRank	21.20	25.31	4.97	6.72
BC + Low-level RL + RPRankNet	23.00	24.62	4.94	6.46
BC + Low-level RL + RPLambdaRank	22.91	22.12	4.85	5.58
HRL	25.38	25.64	6.34	**8.20**
HRL(without state representation module)	22.56	24.10	4.99	7.10

Table 3: Performance comparison with baseline on dataset FedWeb13

Method	NDCG@10	NDCG@20	NDCG-IA@10	NDCG-IA@20
BC + ISRankNet + RPRankNet	27.89	30.32	7.28	8.95
BC + ISLambdaRank + RPLambdaRank	34.73	33.37	9.09	10.01
High-level RL + ISRankNet	32.83	34.77	8.30	10.19
High-level RL + ISLambdaRank	31.36	34.54	7.81	10.03
BC + Low-level RL + RPRankNet	26.75	29.79	6.32	8.39
BC + Low-level RL + RPLambdaRank	29.54	30.42	6.78	8.21
HRL	**40.77**	**38.69**	**10.83**	**12.94**
HRL(without state representation module)	35.53	35.35	8.70	10.77

Table 4: Performance comparison with baseline on dataset FedWeb14

High-level RL + ISLTR:. a three-stage method similar to HRL, which replaces the low-level item selector with LTR method. As well, two LTR methods: RankNet and LambdaRank are used, which denoted as **High-level RL + ISRankNet** and **High-level RL + ISLambdaRank** seperately.

BC + Low-level RL + RPLTR:. a three-stage method similar to HRL, which replaces the high-level vertical selector with other approachs. The vertical selector is responsible for both vertical selection and result presentation.Therefore, The binary classifier is used for vertical selection and L2R is used for result presentation. they are denoted as **BC + Low-level RL + RPRankNet** and **BC + Low-level RL + RPLambdaRank**.

To evaluate the performance of the algorithm, the evaluation metrics are as follows:

NDCG:. Normalized Discounted Cumulative Gain, a common used retrieval metric, which mainly foucs on the relevance of the selected items to the given query.

NDCG-IA:. intent-aware NDCG, an extension on NDCG, where the diversity of page content is also taken into account to minimize the risk of dissatisfication of the average user. It is regarded as the most important one in all the metrics, and it is also used by the TREC FedWeb track.

3.2 Results Comparison

3.2.1 Comparison With Baselines. To illustrate the proposed model performance, we compare it with the baseline models. The results of HRL and baselines in FedWeb13 and FedWeb14 are shown in Table3 and Table4.

In general **HRL** achieves better NDCG-IA than all the baselines in both dataset. Especially in dataset FedWeb14, HRL's NDCG-IA@20 exceeds the traditional pipeline **BC + ISLTR + RPLTR** by 2.75, which shows the efficiency of HRL.

The Comparison between **BC + ISLTR + RPLTR**, **High-levl RL + ISLTR** and **BC + Low-level RL + RPLTR** shows that three-stage method with flat RL can not solve the problem efficiently. Compare to **BC + ISLTR + RPLTR**, **BC + Low-level RL + RPLTR**'s performance has even declined. It is possible that the trained BC for vertical selection limits the exploration of low-level RL, which leads to the degradation of the performance. This illustrates that the performance of different models is similar when solving subtasks. And using the end-to-end HRL model, the three subtasks can be jointly optimized, and the correlation between the subtasks is considered to effectively improve the model performance.

To explore the role of state representation module, we also evaluated the performance of HRL without state representation module. The experimental results show that a well represented state is favorable to RL learning.

As for the gap between model performance between FedWeb13 and FedWeb14 may be due to the difference between the datasets. The queries given by Fedweb13 are more ambiguous compared to Fedweb14, and are related to more vertical. At this point, the low-level RL needs to adapt to more kinds of vertical and the complexity has increased.

3.2.2 Comparison Between Different Training Methods. To illustrate the effect of the training method of the state representation module on the performance of the model, we compared model performance under different training methods. The results of different training methods with different parameter settings are shown in Table 5 and Table 6.

Method	NDCG@10	NDCG@20	NDCG-IA@10	NDCG-IA@20
HRL(without self-supervised learning)	24.26	24.11	5.90	7.35
HRL(alternative training)	24.36	24.62	5.90	7.51
HRL(hybrid loss training, $\beta_{rep} = 0.1$)	23.28	23.54	5.30	6.79
HRL(hybrid loss training, $\beta_{rep} = 1$)	**25.38**	**25.64**	**6.34**	**8.20**
HRL(hybrid loss training, $\beta_{rep} = 10$)	23.36	24.05	5.80	7.55

Table 5: Performance comparison between different training methods on dataset FedWeb13

Method	NDCG@10	NDCG@20	NDCG-IA@10	NDCG-IA@20
HRL(without self-supervised learning)	36.41	35.73	9.50	11.63
HRL(alternative training)	38.07	36.48	9.99	11.94
HRL(hybrid loss training, $\beta_{rep} = 0.1$)	**40.77**	**38.69**	**10.83**	**12.94**
HRL(hybrid loss training, $\beta_{rep} = 1$)	37.16	36.23	9.93	11.97
HRL(hybrid loss training, $\beta_{rep} = 10$)	37.62	36.46	9.90	11.98

Table 6: Performance comparison between different training methods on dataset FedWeb14

Comparing state representation module trained with and without supplementary self-supervised signal, we found that the additional signal did form a better state representation, which helped in RL learning. The best performing model with self-supervised signal outperformed the model without the signal by about 1.0 on NDCG-IA@20 metric for both datasets.

Model trained with hybrid loss function was also found perform better than alternative trained model, which suggests that extrinsic reward is also an important signal to obtain a state representation that assists in the training of the RL agent.

However, the training method with hybrid loss function is sensitive to parameter β_{rep}. The experimental results show that differenet datasets have different requirments for parameter β_{rep}. For Fedweb13, the model performs best with $\beta_{rep} = 1$, but for Fedweb14, $\beta_{rep} = 0.1$ is the best choice. And if the parameters are not chosen properly, the model performance will be greatly affected. For example, in FedWeb13, when $\beta_{rep} = 10$, the performance is even worse than the model without self-supervised learning. In contrast, the advantage of alternative training is no parameter tuning is required and it is more robust. Therefore, different methods can be used in different scenarios.

4 CONCLUSION

In this paper, we formulate the aggregated search problem into a hierarchical end-to-end framework, avoiding training a separate model for each subtask. Meanwhile, we propose an HRL model consisting of two components: a high-level vertical selector for vertical selection and result presentation and a low-level item selector for item selection. A self-supervised learning based state representation method is used to take full account of the correlation between subtasks and enhance the communication between the RL agents. Experimental results on public TREC datasets show that our HRL method outperforms baseline in evaluation metrics.

Limitations of the proposed method may lie in the features and the RL algorithm. In our work, the features are obtained simply by encoding the textual information using doc2vec. Other non-textual information and heuristic features can be added to improve the model performance. Further, the state-of-the-art RL algorithm can also be used within the proposed framework, which is inherently flexible and compatible with a variety of different RL algorithms. In addition, due to the shortage of annotation data in the aggregated search domain, compared to training with full supervision, using a weak supervised or even unsupervised reward signal may be a better choice. Hence, the future research will concern designing reward signals without fully annotated documents and refining the details of the proposed model.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their detailed review and thoughtful suggestions. This work is supported by the NSFC projects (No. 61402403, No. 62072399), Chinese Knowledge Center for Engineering Sciences and Technology, MoE Engineering Research Center of Digital Library, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*. 5–14.

[2] Jaime Arguello, Fernando Diaz, and Jamie Callan. 2011. Learning to aggregate vertical results into web search results. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 201–210.

[3] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. 2009. Sources of evidence for vertical selection. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 315–322.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.

[5] Christopher J Burges, Robert Ragno, and Quoc V Le. 2007. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*. 193–200.

[6] Thomas Demeester, Dolf Trieschnigg, Ke Zhou, Dong Nguyen, and Djoerd Hiemstra. 2015. FedWeb Greatest Hits Presenting the New Test Collection for Federated Web Search. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15)*. ACM, New York, NY, USA.

[7] 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 *Proceedings of the 2018 World Wide Web Conference*. 1939–1948.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Dzung Hong, Luo Si, Paul Bracke, Michael Witt, and Tim Juchcinski. 2010. A joint probabilistic classification model for resource selection. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 98–105.

[10] Xinting Huang, Jianzhong Qi, Yu Sun, Rui Zhang, and Hai-Tao Zheng. 2019. Carl: Aggregated search with context-aware module embedding learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[11] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[12] 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 *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1195–1203.

[13] Or Levi, Ido Guy, Fiana Raiber, and Oren Kurland. 2018. Selective cluster presentation on the search results page. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 1–42.

[14] 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. *arXiv preprint arXiv:1704.03084* (2017).

[15] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[16] 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, 1-2 (1999), 181–211.

[17] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7072–7079.

[18] 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 *The World Wide Web Conference*. 1771–1781.

[19] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 945–948.

[20] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[21] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov decision process for search result diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 535–544.

[22] Hai-Tao Yu. 2020. PT-Ranking: A Benchmarking Platform for Neural Learning-to-Rank. *arXiv preprint arXiv:2008.13368* (2020).

[23] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. 2019. Hierarchical reinforcement learning for course recommendation in moocs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 435–442.