Sequential Evaluation and Generation Framework for Combinatorial Recommender System

Fan Wang*  
Baidu Inc.  
Shenzhen, China  
wang.fan@baidu.com

Xiaomin Fang*  
Baidu Inc.  
Shenzhen, China  
fangxiaomin01@baidu.com

Lihang Liu  
Baidu Inc.  
Shenzhen, China  
liulihang@baidu.com

Yaxue Chen  
Baidu Inc.  
Shenzhen, China  
chenyaxue@baidu.com

Jiucheng Tao  
Baidu Inc.  
Shenzhen, China  
taojiucheng@baidu.com

Zhiming Peng  
Baidu Inc.  
Beijing, China  
pengzhiming01@baidu.com

Cihang Jin  
Baidu Inc.  
Beijing, China  
jincihang@baidu.com

Hao Tian  
Baidu Research  
Sunnyvale, United States  
tianhao@baidu.com

ABSTRACT

Typical recommender systems push K items at once in the result page in the form of a feed, in which the selection and the order of the items are important for user experience. In this paper, we formalize the K-item recommendation problem as taking an unordered set of candidate items as input, and exporting an ordered list of selected items as output. The goal is to maximize the overall utility, e.g. the click through rate, of the whole list. As one solution to the K-item recommendation problem under this proposition, we proposed a new ranking framework called the Evaluator-Generator framework. In this framework, the Evaluator is trained on user logs to precisely predict the expected feedback of each item by fully considering its intra-list correlations with other co-exposed items. On the other hand, the Generator will generate different sequences from which the Evaluator will choose one sequence as the final recommendation. In our experiments, both the offline analysis and the online test show the effectiveness of our proposed framework. Furthermore, we show that the offline behavior of the Evaluator is consistent with the realistic online environment.

1 INTRODUCTION

Recommender Systems(RS) attracts a lot of attention with the booming of information on the Internet. Typical RS algorithms include collaborative methods ([26], [19], [17]), content based methods or hybrid methods of the two([1]). Among those researches, several topics have drawn more attentions in recent years: Context-Aware Recommendation([1]) seeks to model the scenes where the users are feed more precisely; Time-Aware RS([6]) considers both the transformation of the user interest and the change of the value of the news content; Diversified Ranking([31]) focuses on addressing the intra-list correlations.

We focus on the last issue that is related to the arrangement of items in one exposure. It’s important to consider the correlations between the items in many realistic RS in order to maximize the overall utility of the recommended list. While there have been rich resources of studies ([26], [19], [17]) towards improving the precision of recommending a single item at one time, the research on intra-list correlations usually collapses to the problem of diversity ([31]) in many previous works. Yet we think that the investigation in this correlation is far from enough in the several aspects: Firstly, the evaluation of diversity misses a gold standard for a long time. Though there have been different metrics such as Coverage, Intra-List Similarity etc([38], [27]), those evaluations are typically subjective, and not directly related to the user feedbacks. Secondly, the previous works imposed strong propositions on how the items are correlated. Algorithms such as DPP[31] and submodular ranking use hand-crafted kernels or functions for the diversity, they cannot effectively model all possible correlation forms. Thirdly, the traditional step-wise greedy ranking method for the K-item RS cannot guarantee the global optimum. Take the submodular ranking as an example, even if there are no errors in its proposition, the lower bound of the ratio of greedy choice to the global optimum is \(1 - 1/e\) ([33]). Yet, the loss of the local optimum compared with the global optimum is totally unclear for realistic intra-list correlations, when taking into account more kinds of forms of correlations besides the sumodularity.

In this paper, we propose a new solution framework for addressing intra-list correlations: the Evaluator-Generator framework. On the one hand of this framework, the Evaluator aims at fully modeling all possible forms of intra-list interaction. It is trained on user logs to precisely predict the user feedback on each single item conditioned on all its surrounding items. The Evaluator is able to predict the utility of the whole list, such that the global optimum is pursued. In the meantime, the diversity and other requirements from the users will be satisfied to a proper extent when seeking for this target. On the other hand, the Generator is the sequence generating model trained by the user logs or the Evaluator. Then traditional beam search methods or sampling methods will be applied to generate different sequences, from which the Evaluator will choose one sequence as the final recommendation. Our experiments on the offline analysis and the online test demonstrates the consistency of our proposed

*Both authors contributed equally to this research.

Coverage

\(\text{Intra-List Similarity}\)

\(\text{Coverage}\), \(\text{Intra-List Similarity}\) etc([38], [27]), those evaluations are typically subjective, and not directly related to the user feedbacks. Secondly, the previous works imposed strong propositions on how the items are correlated. Algorithms such as DPP[31] and submodular ranking use hand-crafted kernels or functions for the diversity, they cannot effectively model all possible correlation forms. Thirdly, the traditional step-wise greedy ranking method for the K-item RS cannot guarantee the global optimum. Take the submodular ranking as an example, even if there are no errors in its proposition, the lower bound of the ratio of greedy choice to the global optimum is \(1 - 1/e\) ([33]). Yet, the loss of the local optimum compared with the global optimum is totally unclear for realistic intra-list correlations, when taking into account more kinds of forms of correlations besides the sumodularity.

In this paper, we propose a new solution framework for addressing intra-list correlations: the Evaluator-Generator framework. On the one hand of this framework, the Evaluator aims at fully modeling all possible forms of intra-list interaction. It is trained on user logs to precisely predict the user feedback on each single item conditioned on all its surrounding items. The Evaluator is able to predict the utility of the whole list, such that the global optimum is pursued. In the meantime, the diversity and other requirements from the users will be satisfied to a proper extent when seeking for this target. On the other hand, the Generator is the sequence generating model trained by the user logs or the Evaluator. Then traditional beam search methods or sampling methods will be applied to generate different sequences, from which the Evaluator will choose one sequence as the final recommendation. Our experiments on the offline analysis and the online test demonstrates the consistency of our proposed
framework on offline and online environments. In other words, the generated sequences that are preferred by the Evaluator tend to have better performance in our realistic online RS.

There are several contributions in our work:

- In order to fully consider the intra-list correlations in the recommended list, we proposed a generalized Evaluator model which is directly optimized to maximize the the overall utility of the whole list. The requirements like diversity will be satisfied naturally as a side effect.
- We proposed the Evaluator-Generator framework as a practical large-scale recommender system, in order to recommend the best combination of contents to the user given a candidate set of sequences. With such design, our framework is able to pursue the global optimum in the way of improving the Generator or applying heuristic search algorithms.
- We proved that the Evaluator can be used not only as an offline simulator, but also a selector online. The experimental results both offline and online shows that the Evaluator has a very high correlation with the online performance. We also conducted thorough investigations to understand the intra-list correlations. This framework is fully launched online in our system to influence hundreds of millions of online visits every single day.

2 RELATED WORKS

Our work is closely related to the diversified ranking algorithms. Recent works on this topic include the submodularity([32], [2]), graph based methods([37]), and Determinantial Point Process (DPP) ([5], [20], [31]). However, as mentioned above, those previous works either treat diversity as an independent goal (such as Coverage), or impose strong hypothesis on the form of item correlations. E.g., the diversified ranking with predefined submodular functions supposes that diversity are homogeneous on different topics, and independent of the user. DPP and submodular ranking also suppose that co-exposed items always have a negative impact on the local item. In contrast to those propositions, the correlations in the realistic online RS are often found to violate those rules. Our statistics show that some combinations of the items yields better performance than the same single item appearing in other combinations. Other works have also found similar intriguing phenomenons, such as Serpen-tining[34]. The users are found to prefer discontinuous clicks when browsing the list. Thus it is better to spread those high quality items uniformly over the list instead of clustering them on the top positions. Such phenomenon shows the that intra-list correlations is complex and blended with position bias.

The position bias is also an important and practical issue in RS. Click models are widely studied in Information Retrieval systems, such as the Cascade Click Model([11]) and the Dynamic Bayesian Network([7]). It is found that the position bias is not only related to the user’s personal habit, but also related to the layout design etc([9]). Thus click models often need to be considered case-by-case. Few of the previous works have studied the intra-list correlation and the position bias together. Set based recommendation algorithms including submodular ranking and DPP do not consider the position bias at all. In contrast, our framework has naturally addressed the position bias and intra-list correlations altogether.

We also borrow ideas from applying Reinforcement Learning to Combinatorial Optimization(CO) problems. The pointer network [29] is proposed as neural tool for universal CO. [3] further extended pointer-network by using policy gradients for learning. [12] applied Q-Learning to CO problems on graphs. The aforementioned works have been focused on universal CO problems including Travelling Salesman Problem, Minimum Vertex Cover etc.

Some recent works have been addresses long-term rewards in recommender systems. [15] also work on intra-list correlations, which applies policy gradient and Monte Carlo Tree Search(MCTS) to optimize the α-NDCG in diversified ranking for the global optimum. Other works pursues long-term rewards in inter-list recommendations ([35], [36]). Though [35] also proposed treating a page of recommendation list as a whole, the intra-list correlations are not well modelled and analyzed in their work. Also their work is not testified on realistic online RS. In this paper, we more thoroughly investigates the general form of the intra-list correlations. We formalize the utility of the whole list as our final target, but we have also used item-level feedbacks. This has not been sufficiently studied in the above works.

3 BACKGROUNDS

3.1 Problem Setup

We formulate the K-item RS problem as follows: The environment exposes the user profile $u$ and a candidate set $C = \{c_1, c_2, ..., c_N\}$ to the RS, where $N$ is the cardinality of the candidate set and $c_i$ denotes the i-th item. The system picks a recommendation sequence $S = \{c_{s_1}, c_{s_2}, ..., c_{s_K}\}$ where $s_j \in [1, N]$ and $N \geq K$. $S$ is exposed to the user, and the user returns the feedback of $R = \{r_1, r_2, ..., r_K\}$. Furthermore, we denote $S_j^+ = \{c_{s_{j+1}}, c_{s_{j+2}}, ..., c_{s_K}\}$ as the preceeding recommendations above the j-th position, and $S_j^- = \{c_{s_{j-1}}, c_{s_{j-2}}, ..., c_{s_j}\}$ as the recommendations that follows. We define the final objective as maximizing the expected overall utility of the sequence $S$, which is written as $E(R(u, S))$, where the utility $R(u, S)$ is defined by Eq. 1.

$$ R(u, S) = \sum_{j=1}^{K} r_j(u, S). \quad (1) $$

In Eq. 1 we assume that $r_j(u, S)$ is not only related to the user and item $c_{s_j}$ in position $j$, but also related to its preceding and following items $S_j^+$ and $S_j^-$. Also notice that we made no proposition on the correlation, which means there are not only possibilities of negative and positive correlations, but also higher order correlations (such as correlation of three items). This is essentially different from many previous proposed diversified ranking algorithms.

3.2 Item Independent Ranking

Most of the previous RS are based on step-by-step greedy ranking. It goes one-by-one without considering the item correlation, which is written as

$$ f_0(u, j, s_j) \rightarrow E(S_j^+, S_j^-)(r_j(u, S)), \quad s_j = \arg\max_i \{f_0(u, j, i)\}. \quad (2) $$

Wang and Fang, et al.
We use the formulation $A \rightarrow B$ here to represent that "A is to approximate B". The parameterized function $f_\theta$ with trainable parameter $\theta$ is optimized to approximate the click through rate (CTR) $E_{S_j^+}(r_j(u, S))$, the subscript here means that the item $c_{ij}$ could appear with any $S_j^-$ and $S_j^+$, and Eq. 2 simply averaged out all that information. We named $f_\theta(u, j, s_j)$ the Item Independent Prediction (IIP).

### 3.3 Preceding Items Aware Prediction

It is natural to improve Eq. 2 by taking $S_j^+$ into account. We define the Preceding Items Aware Prediction (PIAP) as Eq. 3.

$$f_\theta(u, j, S_j^-) \rightarrow E_{S_j^+}(r_j(u, S)),$$

$$s_j = \text{argmax}_i (f_\theta(u, i, S_j^-)), S_{j+1}^- = S_j^- \cup c_{sj}. \quad (3)$$

Here we omit the position $j$ in $f_\theta(u, j, s_j, S_j^-)$ and simply write $f_\theta(u, j, s_j, S_j^-)$, because the position can be recovered from $S_j^-$, and thus encoding $S_j^-$ automatically incorporates the position bias. A lot of previous diversified recommendation algorithms can be considered as specific paradigm of Eq. 3, e.g., the linear sub-modular recommendation(2). However, we have not made any assumption here about the formula of $f_\theta(u, i, S_j^-)$ yet. We argue that the expression of Eq. 3 essentially accounts for all kinds of correlations, including positive, negative and higher-order correlations.

However, two defects of Eq. 3 need to be mentioned. Firstly, one-by-one ranking paradigm faces the issue of local optimum and the performance of the sequence as a whole is not guaranteed. Secondly, the following items $S_j^+$ is not taken into account, but results has been found to reveal that $S_j^+$ does have an impact in the overall performance of $j$ position, including click models(30) and mouse tracking studies(13). However, knowing $S_j^+$ is not feasible in the ranking process. In the next part we introduce the Evaluator-Generator framework which solves the two issues properly.

## 4 EVALUATOR-GENERATOR FRAMEWORK

### 4.1 The Framework Overall

In order to fully catch the intra-list correlations and the global optimum, we propose to separate the list generation with the evaluation process. The evaluation process considers only about predicting the value of a known sequence, which we call the Evaluator. The generative process is responsible for generating high-value sequences, which we call the Generator. During the inference process, it is possible to use multiple generators to generate multiple lists, from which the Evaluator selects the best one(Fig. 1). We argue that this workflow is beneficial in the following aspects.

- **Thorough consideration of intra-list correlations.** An Evaluator encodes the information of the whole sequence together, the intra-list correlation and position bias is incorporated effectively. It thus achieves higher accuracy compared with either IIP or PIAP in item-level.
- **Light-Weighted Generators.** Limitations on the generator exist as the generative process do not have full access to the sequences information. The Evaluator-Generator allows generating multiple lists from which to choose the best one. It is able to use light-weighted evaluators to explore high-quality candidate lists.
- **Global optimum is ensured.** The Evaluator pursues global optimum explicitly by directly comparing the utility of the whole list. E.g., by encoding the whole sequence $S$, the utility score not only encourage items that achieve higher click-through-rate itself, but also encourage items that attract following examination and clicks to the other items in the result page.

### 4.2 The Evaluator

#### 4.2.1 Contextual Items Aware Prediction

We propose the Contextual Items Aware Prediction (CIAP), which encodes both $S_j^-$ and $S_j^+$ to predict the feedback on $j$th position, which is written as.

$$f_\theta(u, j, S_j^-, S_j^+) \rightarrow E(r_j(u, S)). \quad (4)$$

We sum up the predictions at each single position for a sequence as an estimation of $R(u, S)$ all-together, that is,

$$\hat{R}(u, S) = \sum_j f_\theta(u, j, S_j^-, S_j^+). \quad (5)$$

We can see that $\hat{R}(u, S) \leftarrow E(R(u, S))$ is an approximation to the expectation of Eq. 1. Given $n$ sequences $\{S_1, S_2, ..., S_n\}$, the Evaluator can be used to select the sequence greedily by $S = \text{argmax}_s \hat{R}(u, S)$.

Again, encoding the whole sequence automatically handles position bias as well. Also, the user feedback in each single position is utilized. Compared with complete page-wise regressions, Eq. 4 and Eq. 5 keeps more item-level information.

#### 4.2.2 Determinantal Point Process

Among all previous works, Determinant Point Process (DPP, [20]) and Submodular ranking([4]) is closely related to our work. We briefly review the DPP models for RS. DPP models the probability of a subset being clicked as $P(R)$. It explicitly set the correlation to be $K_{i,j} = K(c_{ix}, c_{jy})$, where $K(x, y)$ is the kernel function that maps $(x, y)$ to a real value symmetrically. Assuming that $Y = \{i \in [1, K] | r_i = 1\}$, and $K_Y$ represent the sub-matrix of $K$ with only the index of rows and columns in $Y$. DPP proposes that $P(R)$ obeys the following rule:

$$P(R) \propto \frac{|K_Y|}{|K + I|}. \quad (6)$$

With minor mathematical calculations it is shown that the function $P()$ in Eq. 6 satisfies submodularity. [31] proposed a practical framework for deploying DPP to diversified ranking(Deep DPP). Each element of $K_{i,j}$ is decomposed to a multiplication of quality term and a correlation term, both of which was calculated separately from a deep neural network.

DPP can be interpreted as one special type of Evaluator which estimate the value based on a full set(or sequence) instead of a single item. Though DPP only models the set-wise click probability $P()$, it is not strictly correlated to the sum of clicks $R(u, S)$. However, DPP maximizes $|K_S|$ in its ranking process, just like Evaluator maximizing $\hat{R}(u, S)$, thus we treat $|K_S|$ as the evaluation score in the experiments.
4.3 The Generators

We denote the Generator policy as $\pi_\theta(S; u, C)$ (with slight abuse of notation, $\theta$ is again the parameter for the generator) that represents the possibility of generating the $S$ given the user $u$ and the candidates $C$. Though various paradigm to generate a list exist, here we mainly use sequential decision policy, which is defined as Eq. 7.

$$\pi(S; u, C) = \prod_{j=1}^{K} p_\theta(s_j|u, S_j^+, C)$$  \hfill (7)

4.3.1 Softmax Sampling. The simplest way is to use some heuristic values to generate multiple sequences by applying softmax sampling, in which the decision probability $p_\theta(s_j|u, S_j^+, C)$ is written as

$$p_\theta(s_j|u, S_j^+, C) = \exp(f_\theta(u, s_j, S_j^+))/\tau$$  \hfill (8)

We call the heuristic value $f_\theta(u, s_j, S_j^+)$ the priority score. It is straightforward to use IIP or PIAP as $f_\theta$, results in IIP + Softmax and PIAP + Softmax policies. Also notice that if $\tau \to 0$ Eq. 8 degenerate to greedy policy.

4.3.2 Reinforcement Learning. It is possible to interpret Eq. 1 as a Combinatorial Optimization (CO) problem with stochastic evaluation. The pointer-network and Reinforcement Learning approach to solve CO has been found to be promising recently by [29] and [3]. We formulate the generation of the ranking list as Markov Decision Process, which is described as follows: At each step $j$, the state $x_j$ is decided by the user profile $u$, the index of the items that has been chosen $S_{j+1}$, and the candidates $C$, i.e $x_j = [u, S_j^+, C]$. The policy is represented by $p_\theta(s_j|u, S_j^+, C)$ similarly as Eq. 8. Each time after choosing the item at $j$-th step, a new state $x_{j+1} = [u, S_{j+1}, C]$ and a reward $r_j$ is exposed.

As we know that vanilla policy gradient often suffers from large variance and low data efficiency. Recently the sample efficient Q-learning has been applied to combinatorial optimization problem([18]). We denote the value function of state-action pair $(x_j, i)$ that follows the generative policy $\pi$ as $Q^\pi(x_j, i)$, which is written as

$$Q^\pi(x_j, i) = E_{s_j=i, s_{j+1}, s_{j+2}, \ldots, s_K \sim \pi}(r_j + r_{j+1} + \ldots r_K).$$  \hfill (9)

We use the value function approximation $f_\theta(u, s_j, S_j^+, C)$ to approximate the value function of the optimal policy, $Q^*(x_j, i)$. The aforementioned IIP, PIAP model here can be used as the function approximator. The Q-Learning minimizes the loss in Eq. 10.

$$L_{td}(\theta) = \sum_{S} \sum_{i} [\max_{j} \{f_\theta(u, i, S_{j+1}, C)\} + r_j - f_\theta(u, s_j, S_j^+, C)]^2.$$  \hfill (10)

$\theta^*$ is the parameters of the target model that follows $\theta$ with delay. Also notice that the discount factor of reinforcement learning $\gamma = 1$ in our problem. A detailed explanation of Deep Q Learning can be found in [25]. Q-learning is an Off-Policy learning metric, thus we can use user interactions offline to train.

4.3.3 Reinforcement Learning with Virtual Reward. As RL often requires large amount of data, we propose using the Evaluator as an offline estimator and reward function, i.e. replacing the real user feedback $R$ with the estimated rewards of $f_\theta$ in Eq. 5. This is similar to the counterfactual estimators([16]) and the reward model([21]). We call it learning from Virtual Reward. There are risks to directly learn from the offline estimator: Neural networks are known to be vulnerable to adversarial attacks([23]), while the learning process of Generator resembles black box adversarial attack. Once there is discrepancy between the Evaluator and the realistic environment, the Generator would deviate from the real target.

However, learning from the Evaluators is more data efficient. In our offline experiments, we also evaluate the Generator using the Evaluator as the gold standard. In other words, we treat the Evaluator as a simulator(or Environment). We also carried out online experiments on the virtual reward trained Generator in order to test its validity. We will come to this again in the experiments section.

5 MODEL ARCHITECTURES

Notice that until now we have made no proposition on the paradigm of the policy or prediction model. There are whole bunches of neural network models that proposed before to tackle different aspects of RS, including the well known Youtube RS([10]), Google deep-and-wide RS([8]). The model structure is highly coupled with the problem and the features. In this paper, in order to compare the framework itself, we propose several simple and comparable different model architectures. It is worth mentioning that we can surely add more complexity to those structures to account for more features and factors (such as dynamic user interest), but those are not the main focus of this paper.

DNN for IIP. Considering Eq. 2, where $f_\theta(u, j, s_j)$ is only related to user, item and its position, we use Multi-Layer Perceptrons(MLP) to build the mapping. We represent the user with embeddings $\phi(u)$, item with $\phi(c_j)$, and position with $\phi(j)$. In our dataset, the representation of item is the concatenation of the item id embedding and other features (such as categories and titles etc). We use two layer MLP
After encoding the preceding items, we concatenate the representation between each pair of items as proposed by [31]. A MLP header is

\[ f_0(u, s_j) = MLP_2(\phi(u), \phi(c_s), \phi(j)). \quad (11) \]

GRNN for PIAP. To model the context items \( S_j^- \), we use recurrent neural structures (such as Gated Recurrent Neural Network, GRNN). After encoding the preceding items, we concatenate the representation of \( u, c_s, S_j^- \), and apply the two-layer MLP (Fig. 2b). The model can be written as

\[
\begin{align*}
\mathbf{h}_j^* &= GRU(h_{j+1}^- \cdot \phi(c_s)), \\
f_0(u, j, S_j^-) &= MLP_2(\phi(u), \phi(c_s), \mathbf{h}_j^*). \quad (12)
\end{align*}
\]

Notice that position \( j \) does not appear explicitly in the GRNN model. We require the model to encode the position implicitly by forward passing of GRU.

Bi-GRNN for CIAP. For the Evaluators, to fully take \( S_j^+ \) into account, we apply an second GRNN in reversed direction, followed by the same two-layer MLP (Fig. 2c), which we call Bi-directional (24) GRNN (Bi-GRNN).

\[
\begin{align*}
\mathbf{h}_j^+ &= GRU(h_{j-1}^+ \cdot \phi(c_s)), \\
\mathbf{h}_j^* &= GRU(h_{j+1}^+ \cdot \phi(c_s)), \\
f_0(u, j, S_j^+; S_j^-) &= MLP_2(\phi(u), \phi(c_s), \mathbf{h}_j^+, \mathbf{h}_j^*). \quad (13)
\end{align*}
\]

Transformer for CIAP. Transformer is proposed by [28] for neural machine translation. Recently there has been great amount of successes in using Transformer to encode sequences. We use Transformer as alternative structures in place of the Bi-GRNN. We first concatenate the user descriptor with item representation and position embedding, and then we apply 2-layer Transformer to predict the probability of click in each position. The details are shown in Eq. 14. In our attention models, we use single head attentions.

\[
\begin{align*}
e_j &= (\phi(u), \phi(c_s), \phi(j)), \\
f_0(u, s_j, S_j^+) &= \text{Transformer}_2(e_1, e_2, \ldots e_K). \quad (14)
\end{align*}
\]

Deep DPP. We use the similar kernel to model the correlation between each pair of items as proposed by [31]. A MLP header is used to predict the quality term \( q(u, i) \) for user \( u \) and item \( c_i \) and a kernel matrix \( K \) is introduced to represent the correlations.

\[
\begin{align*}
q(u, i) &= MLP_2(\phi(u), \phi(c_i)), \\
K_{i,j}(u, S) &= \exp(-D_{s_i, s_j}), \quad (15)
\end{align*}
\]

where \( D_{s_i, s_j} \) are the Jaccard distances of item \( c_i \) and item \( c_j \).

**Pointer Network** [29]. Pointer Network is used to copy and readjust the order of the input sequence. Here we use Pointer Network to generate sequence \( S \) from the original unordered set \( C \). The pointer network encodes the candidate set \( C \) using recursive structure, and applies a decoder to generate the sequence \( S \). We write the generation process as

\[
\begin{align*}
\mathbf{h}_j^+ &= GRU(h_{j-1}^+ \cdot \phi(c_i)), \\
\mathbf{h}_0 &= MLP(h_{N+1}^+ \cdot \phi(u)), \\
\mathbf{h}_j^* &= GRU(h_{j-1}^+ \cdot \phi(c_s)), \\
f_0(u, s_j, S_j^+, C) &= MLP_2(\phi(u), \mathbf{h}_j^*, \phi(c_s)), \\
p_{\theta}(s_j | u, S_j^+, C) &= \exp(f_0(u, s_j, S_j^+, C) / \tau).
\end{align*}
\]

The vanilla pointer network outputs the attention over candidates \( p_{\theta} \). Here when trained with Q-Learning, we use \( f_0 \) to approximate the Q value, and then we use \( p_{\theta} \) to inference. Moreover, the vanilla pointer network is not devised to take the other context information (such as user \( u \)) into account. To improve this we inject the user feature at both the end of the encoding process and each decoding step, in order for the pointer network to address the user preference thoroughly.

### 6 EXPERIMENTS

In our experiments, we use the user interaction records of 100 million lists for training and 1 million lists for testing, which were collected from Baidu App News Feed System (BANFS), one of the largest RS in China. BANFS has over hundreds of millions of clicks online each day. The length of each sequence may be different. To reduce the cost of the experiment, our offline dataset \(^1\) contains only a subset of the features, including the user id, item id, item category and layout (the appearance of the item).

\(^1\)The dataset is not yet ready to be published due to the privacy issues. In case the data is fully desentizated, publishing the dataset would be considered.
6.1 Evaluation Metrics

Traditional IR evaluation metrics (such as NDCG, MAP, ERR) is not suitable for evaluation of intra-list correlations. Firstly, those metrics make the assumption that user feedback are fully observable for all the items, or there are oracle rankings, but it is no longer true for interactive systems such as news feed RS. Secondly, those metrics assume that an item is independent to each other.

To evaluate combinatorial RS, [33] uses an artificial simulator, others[31] uses online experiments for evaluation. Also counterfactual evaluation tricks has been reported([14],[22]), but applying those methods to realistic RS with millions of candidate items and billions of users are often intractable.

In this work, we evaluate our ideas from the following aspects.

- Firstly, three metrics are exploited to evaluate the precision of Evaluator through realistic user feedback. Metric Area Under Curve(AUC) of the ROC curve are used to evaluate the precision of each item in each sequence. RMSES and Correlation are used to evaluate the overall click \( E(R(u, S)) \) of a sequence. Root Mean Square Error of Sequences(RMSES) is defined as

\[
RMSES = \sqrt{\frac{1}{n} \sum_i (\bar{R}(u, S) - R(u, S))^2}
\]

Since some methods, such as DPP, do not predict \( R(u, S) \) in the right scale, we also evaluate the Correlation between total real clicks \( R(u, S) \) and the evaluation score \( \bar{R}(u, S) \), which is defined in Eq. 18.

\[
\text{Correlation} = \frac{\text{Cov}(\bar{R}(u, S), R(u, S))}{\sqrt{\text{Var}(\bar{R}(u, S)) \cdot \text{Var}(R(u, S))}}
\]

- Secondly, we compare different Generators by regarding the Evaluator itself as an simulator(or environment). We also publish some of our research on the intuitive perception of the generated patterns.

- Finally, we publish the result to compare different ranking frameworks in online A/B tests. As we are more cautious toward online experiments, we did not carry out the experiments on all possible ranking frameworks. It is worth noticing that our online experiments uses larger feature set and datasets to achieve better performance, thus the performance of the online and offline experiments are not totally equal.

6.2 Results on The Evaluators

To evaluate the precision of the Evaluator models, we use AUC, RMSES and Correlation as the criteria for comparison. We compare different models including DNN(IIP), GRNN(PIAP), Bi-GRNN(CIAP), Transformer(CIAP) and DPP. By concluding from Tab. 1, we can see that \( S_j^- \) and \( S_j^+ \) do have impact on the click of the \( j \)-th position; The Bi-GRNN and Transformer performs best in all three evaluation criteria; Transformer is slightly better than Bi-GRNN, as expected; The performance of DPP is below the baseline, which is mainly caused by missing the position bias. To better study the impact of context items, we further replace all the preceding items in \( S_j^- \) or the following items in \( S_j^+ \) randomly in Bi-GRNN and Transformer. The performance is shown as the Bi-GRNN(Disturb) and Transformer(Disturb) in Tab. 1.

![Table 1](image)

### Table 1: Comparison of the AUC, RMSES and Correlation for different evaluators

| Algorithm       | AUC  | RMSES | Correlation |
|-----------------|------|-------|-------------|
| DNN(IIP)        | 0.7658 | 0.2781 | 0.4685      |
| GRNN(PIAP)      | 0.7730 | 0.2769 | 0.4838      |
| Bi-GRNN(CIAP)   | 0.7793 | 0.2760 | 0.4925      |
| Bi-GRNN(Disturb \( S_j^- \)) | 0.7708 | -     | -           |
| Bi-GRNN(Disturb \( S_j^+ \)) | 0.7714 | -     | -           |
| Transformer(CIAP)| 0.7802 | 0.2760 | 0.4969      |
| Transformer(Disturb \( S_j^- \)) | 0.7719 | -     | -           |
| Transformer(Disturb \( S_j^+ \)) | 0.7738 | -     | -           |
| Deep DPP        | -    | -     | 0.3810      |

To better study this phenomenon, we further plot the correlation of clicks between any two positions. Fig. 3c shows that the correlation of user clicks is interweaving: the adjacent positions is less likely to be clicked together, but \( j \) and \( j + 2 \) is more likely to be clicked together. This is in consistency with the Serpentine phenomenon that was mentioned by [34]. This phenomenon has further shown that the intra-list correlation is much more complicated than many position bias hypothesis or unordered set-wise hypothesis previously proposed.

6.3 Results on the Generators

To evaluate the Generators, we randomly sample 1 million candidate sets from the user interaction logs. Those are regarded as pools of candidate \( C \). In each iteration, we randomly sample an user \( u \), a candidate set \( C \) and length of the final list \( K \). The length \( K \) follows the real distribution of sequence length online, which varies between 10 and 50. We sample the candidate set such that \( N = 2K \). Then, the Generator is required to generate one or multiple sequences of length \( K \), and we use the Evaluator to pick the sequence with highest overall score (if only 1 list is generated, such as greedy picking, then there are no need of Evaluators). We compare the evaluation score of the finally picked list. There are mainly three types of Generators to be compared:

- **Supervised Learning(SL) + Greedy** The Generator is trained with normal log-likelihood loss, using user feedbacks in the log. We generate the list with greedy policy as classical ranking process.
The result is shown in Tab. 2. Several remarks can be made:

- **SL + Softmax** We use the predicted CTR score as the priority score, and then we apply softmax sampling of Eq. 8, generating \( n \) list with temperature of \( r = 0.002 \). Then, we use the corresponding Evaluator to pick the one with highest score.
- **Reinforcement Learning (RL) with Virtual Rewards** We apply RL to the Generator using reward from the corresponding Evaluators. When testing, the Generator generates only one list greedily.

We choose three Evaluators, GRNN, Bi-GRNN and Transformer. The result is shown in Tab. 2. Several remarks can be made: **GRNN + SL + Greedy** can not compare **GRNN + SL + Softmax** (100 Sample), even if the Evaluator is itself (GRNN Evaluator). This investigation shows that the difference between local and global optimum is considerably large. As we expected, **RL + Greedy** outperforms **SL + Greedy**, since it learns more combinatorial information from Evaluator. Additionally, we can see that **Softmax (100 Sample)** achieves the best performance under all 3 Evaluators, which indicates that it is hard for a single sequence to beat multiple different sequences, even though that sequence is generated by a strong Generator, such as **PointerNet + RL**.

To illustrate that the Evaluator-Generator framework indeed yields better item combinations, we did some inspection in the generated patterns. The combination of item ids or high dimensional features are far too sparse to give any insightful results, thus we focus on the layouts of the short local segments. The layout marks the visual content arrangement within each item when shown to the users, as the layouts of the short local segments. The layout marks the visual qualities of the pattern, we use log. Under the assumption that the higher click rate means better quality of the pattern, we use \( R(P_{l_i, l_j, l_k}) \) to measure the quality of the local layout pattern \( P_{l_i, l_j, l_k} \). So we regard the layout patterns in 216 possible patterns that rank top-N in the expected clicks as “good patterns”. To evaluate our proposed framework, we calculate the ratio of the top-N pattern segments in the generated lists from different Generators. The results are shown in Tab. 3, where we use Bi-GRNN as the Evaluator. Comparing with Tab. 2, the Softmax and the RL group generates more “good patterns”, as well as score higher in the Evaluator, compared to the SL+Greedy methods. This results demonstrate that our proposed Evaluator-Generator framework is consistent with that kind of intuitive indicators, i.e., higher score RS generates more patterns that are preferred to be clicked by the users.

### 6.4 Performance Online

**Correlation between Evaluators and Online-Performance** The previous results show that the Evaluator is more correlated to the sum of clicks of a list. But, is the predicted sum of clicks related to the final performance? Is it appropriate to treat Evaluator as a simulator? We perform additional investigation on the correlation between the Evaluation score of lists \( R(u, S) \) and the performance of A/B test. Typically we judge whether a new policy is superior than the baseline, by the increase in some critical metrics, such as total user clicks, in A/B test. For two experiment groups with experiment ID \( I_A \) (experimental) and \( I_B \) (baseline), the **Relative Gain** in total clicks is defined as the relative increase of clicks compared with baseline. Thus we retrieve the logs of the past experiments, and we re-predict click of each sequences in the record by inferencing with our Evaluator model. We calculate the **Predicted Relative Gain** by

\[
\Delta = \frac{\sum_{i \in I_A} \hat{R}(u, S_i) - \sum_{i \in I_B} \hat{R}(u, S_i)}{\sum_{i \in I_B} \hat{R}(u, S_i)}. \tag{19}
\]

We collect over 400 A/B testing experiments during 2018, including not only new ranking strategies with various policy, model and new features, but also human rules. Some of the new strategies are tested to be positive, others negative. We counted the correlation between the predicted relative gain and the statistical real relative gain. We use DNN and Bi-GRNN Evaluator for comparison. The correlation between DNN(IIP) and online performance among the 400 experiments is 0.2282, while the correlation between Bi-GRNN(CIAP) and real performance is as high as 0.9680. It has
TABLE 2: COMPARISON OF DIFFERENT LIST GENERATORS

| Generator                        | GRNN | Bi-GRNN | Transformer |
|----------------------------------|------|---------|-------------|
| DNN + SL + Greedy                | -    | 1.1416  | 1.5887      |
| GRNN + SL + Greedy               | 1.7162 | 1.5322  | 1.7019      |
| GRNN + SL + Softmax \((n = 20)\) | 1.8134 | 1.7429  | 1.8368      |
| GRNN + SL + Softmax \((n = 100)\) | 1.8676 | 1.8179  | 1.9032      |
| PointerNet + RL(Virtual Reward) + Greedy | -    | 1.6372  | 1.8967      |

TABLE 3: STATISTICS OF PROBABILITY OF GENERATING THE TOP-N CLICK RATE COMBINATION OF LAYOUT PATTERNS IN EVALUATOR-GENERATOR FRAMEWORK WITH DIFFERENT GENERATORS AND BI-GRNN EVALUATOR.

| Generator                        | Top-3 Pattern(%) | Top-5 Patterns(%) | Top-10 Patterns(%) |
|----------------------------------|------------------|-------------------|--------------------|
| DNN + SL + Greedy                | 1.46             | 1.85              | 3.43               |
| GRNN + SL + Greedy               | 1.43             | 2.43              | 7.32               |
| DNN + SL + Softmax \((n = 100)\) | 1.31             | 1.72              | 4.50               |
| GRNN + SL + Softmax \((n = 100)\) | 1.80             | 2.88              | 8.54               |
| PointerNet + RL(Virtual Reward) + Greedy | 2.45 | 2.92 | 6.67 |

TABLE 4: CORRELATION OF EVALUATOR PREDICTIONS AND ONLINE A/B TEST PERFORMANCE

| Generator                        | DNN | Bi-GRNN |
|----------------------------------|-----|---------|
| Correlation with Online Performance | 0.2282 | 0.9680 |

7 DISCUSSION

In this paper, we propose a recommender framework by optimizing K-item in the result page as a whole. We propose the Evaluator-Generator framework to solve the combinatorial optimization problem. We show that compared with the other diversified ranking algorithms, the proposed framework is capable of capturing various possible correlations as well as the position bias. In this section, we post some of our further comments in this framework and its possible future extensions.

Exploration and Exploitation. Exploration is very important for interactive systems. Greedy ranking in RS typically ends up in mediocre or outdated recommendations and eventually will jeopardize the performance. Exploration for sequence-optimization RS include not only the item-level, but also the combination level. To explain the second level, if we consider that the user always see a perfect combination every time, the evaluator would not be able to learn that lacking diversity would do harm to the performance. We have also observed the phenomenon online. BANFS keeps a small fraction of flow for exploration.

Evaluator Generator vs RL only Some might argue that a well-designed RL Generator only will be enough for intra-list correlation, where the Evaluator seems redundant. In our case, the Evaluator works as both the offline estimator and online selector. The Evaluator-Generator framework does not only have much more flexibility in realistic online system, it is also close to the “reward modeling” that was proposed in [21]. The Evaluator is able to approximate real user preference better with less effort compared with the Generator. We think that the difference between Generator-Only and Evaluator-Generator deserves further investigation.
Table 5: Comparison of the performance online.

| Recommender Framework                             | Relative Gain (%) | Coverage of Categories(%) |
|----------------------------------------------------|-------------------|---------------------------|
| PIAP vs. IIP                                       | +1.75             | +4.05                     |
| Evaluator-Generator vs. PIAP                        | +2.44             | +0.62                     |
| PointerNet + RL(Virtual Reward) vs. Evaluator-Generator | -0.01            | -                          |
| PointerNet + RL(Real Click) vs. Evaluator-Generator | -0.05             | -                          |

Synthesising Intra-list Correlation and Inter-list Evolution. Though there are some works that shed light into the “ultimate” RS that incorporate Intra-list and Inter-list correlations[35], incorporating both effects in a real RS remains very tricky. Few has been testified in real online systems. We believe that it is a challenging but promising topic in the future.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Recommender systems handbook, pages 191–226. Springer, 2015.

[2] Yossi Azar and Itah Gamzu. Ranking with submodular valuations. In Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms, pages 1070–1079. SIAM, 2011.

[3] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940, 2016.

[4] Allan Borodin, Huyun Chul Lee, and Yu Li. Max-sum diversification, monotone submodular functions and dynamic updates. In Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems, pages 155–166. ACM, 2012.

[5] Alexei Borodin. Determinantal point processes. arXiv preprint arXiv:0911.1153, 2009.

[6] Pedro O Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Modeling and User-Adapted Interaction, 24(1-2):67–119, 2014.

[7] Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In Proceedings of the 18th international conference on World wide web, pages 1–10. ACM, 2009.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pages 7–10. ACM, 2016.

[9] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. Synthesis Lectures on Information Concepts, Retrieval, and Services, 3(7):1–115, 2015.

[10] Paul Cowgill, Joseph A. Konstan. Toward better interactions in recommender systems: cycling and serpentining approaches for top-n item lists. In CSCW2017, pages 257–297. Springer, 2011.

[11] Nick Crawwell, Onno Zoeter, Michael Taylor, and Bill Ramsay. An experimental comparison of click-position-bias models. In Proceedings of the 2008 international conference on web search and data mining, pages 87–94. ACM, 2008.

[12] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. arXiv preprint arXiv:1704.01665, 2017.

[13] Fernando Diaz, Ryen White, Georg Buscher, and Dan Liebling. Robust models of mouse movement on dynamic web search results pages. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 1451–1460. ACM, 2013.

[14] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. arXiv preprint arXiv:1103.4601, 2011.

[15] Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. From greedy selection to exploratory decision-making: Diverse ranking with policy-value networks. 2018.

[16] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. Offline ab testing for recommender systems. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pages 198–206. ACM, 2018.

[17] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 9th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 549–558. ACM, 2016.

[18] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems, pages 6351–6361, 2017.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8), 2009.

[20] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. Foundations and Trends® in Machine Learning, 5(2–3):123–286, 2012.

[21] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. arXiv preprint arXiv:1811.07871, 2018.

[22] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In Proceedings of the 26th International Conference on World Wide Web, pages 929–934. ACM, 2015.

[23] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. arXiv preprint arXiv:1701.06748, 2017.

[24] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01554, 2016.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, pages 285–295. ACM, 2001.

[27] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Recommender systems handbook, pages 257–297. Springer, 2011.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017.

[29] Oriol Vinyals, Meir Fortuny, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700, 2015.

[30] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. Incorporating non-sequential behavior into click models. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 283–292. ACM, 2013.

[31] Mark Wilhelmen, Ayith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. Practical diversified recommendations on youtube with determinantal point processes. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 2165–2173. ACM, 2018.

[32] Yan Yan, Gaowen Liu, Sen Wang, Jian Zhang, and Kai Zheng. Graph-based clustering and ranking for diversified image search. Multimedia Systems, 23(1):41–52, 2017.

[33] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In Advances in Neural Information Processing Systems, pages 2483–2491, 2011.

[34] Qian Zhao, Gediminas Adomavicius, F Maxwell Harper, Martijn Willemsen, and Joseph A Konstan. Toward better interactions in recommender systems: cycling and serpentining approaches for top-n item lists. In CIKM2017, Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, 2017.

[35] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoyue Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. arXiv preprint arXiv:1805.02341, 2018.

[36] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Dn: A deep end-to-end learning framework for news recommendations. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pages 167–176. International World Wide Web Conferences Steering Committee, 2018.

[37] Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. im2vec: Learning discriminative top-down image representations. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017.