AliISA: Creating an Interactive Search Experience in E-commerce Platforms

Fei Xiao, Zhen Wang, Haikuan Huang,
Jun Huang, Xi Chen, Hongbo Deng, Minghui Qiu
{guren.xf,jones.wz,haikuan.hhk,huangjun.hj,gongda.cx,
dhb167148,minghui.qmh}@alibaba-inc.com
Alibaba Group
Hangzhou, China

Xiaoli Gong
gongxiaoli@nankai.edu.cn
Nankai University
Tianjin, China

ABSTRACT

Online shopping has been a habit of more and more people, while most users are unable to craft an informative query, and thus it often takes a long search session to satisfy their purchase intents. We present AliISA—a shopping assistant which offers users some tips to further specify their queries during a search session. With such an interactive search, users tend to find targeted items with fewer page requests, which often means a better user experience. Currently, AliISA assists tens of millions of users per day, earns more usage than existing systems, and consequently brings in a 5% improvement in CVR. In this paper, we present our system, describe the underlying techniques, and discuss our experience in stabilizing reinforcement learning under an E-commerce environment.

CCS CONCEPTS

• **Theory of computation → Reinforcement learning**; **Sequential decision making**.

KEYWORDS

information retrieval, reinforcement learning

ACM Reference Format:
Fei Xiao, Zhen Wang, Haikuan Huang, Jun Huang, Xi Chen, Hongbo Deng, Minghui Qiu and Xiaoli Gong. 2019. AliISA: Creating an Interactive Search Experience in E-commerce Platforms. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331409

1 INTRODUCTION

In recent years, E-commerce (EC) has grew to share more than 10% and 18% retail sales in U.S.A. and China respectively. The head EC platforms serve tens of millions of users everyday with billions of items. Without an informative query, users often suffer from browsing tens of SEarch Result Pages (SERP) before reaching their targeted items. Besides, user intents are usually very vague and broad at the beginning of a search session and become clear and focused, as users browse more and more items.

This raises a need for an interactive search experience that allows users to further specify their queries during the whole search session. Fortunately, each item belongs to some categories (e.g., AirPods belongs to earphone) where items of a certain category can be characterized by a set of category-specific properties (e.g., brand, way of wearing, price range, etc. for earphone). An effective way to narrow down the scope of retrieval is to specify which values do the intended items take over those properties, e.g., "brand is Apple and way of wearing is in-ear" strongly indicates AirPods.

Nowadays, most EC platforms provide a kind of panels (see Fig. 1a) where users are allowed to specify the value(s) of any related property. However, it is infeasible to embed such a huge panel into the SERP, and thus only a small portion of users have found the entrance of the panel. Another kind of shopping assistant is to display a tip in the SERP which consists of several terms that might be suitable for describing user intends (see Fig. 1b). Once the user clicks a certain term, the system would augment the query by it. As the displayed terms may describe different aspects of an item, they often overwhelm the users. Besides, several synonyms often appear in a tip simultaneously which also degenerates user experience.

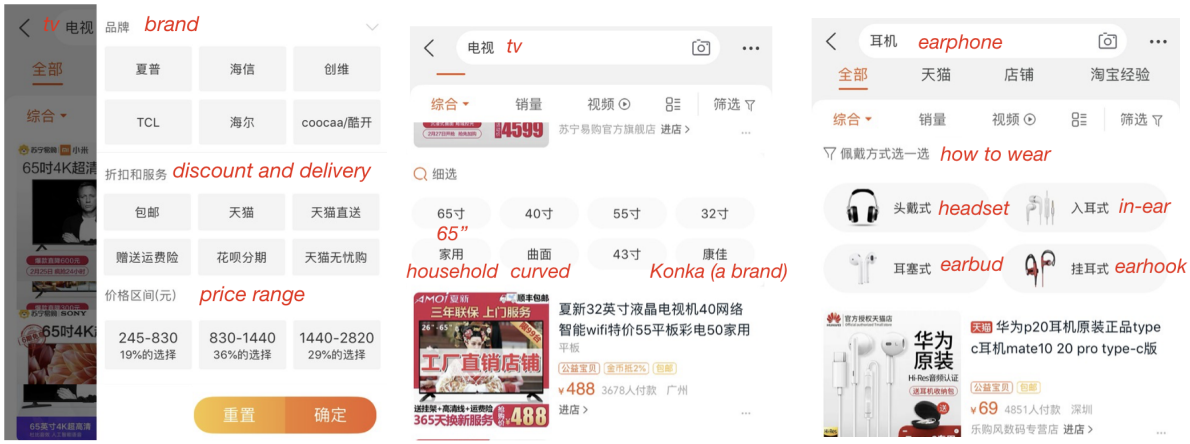
In this paper, we present AliISA (/ˈɑliːʃɑ/)—a shopping assistant which, in each SERP, offers our users a certain property (e.g., way of wearing) and its values (see Fig. 1c). Users may click a certain value (e.g., headset) that best describes their intention. In response, AliISA rewrite the query (e.g., from 'earphone' to 'headset earphone') for requesting the next SERP. With such an interactive search, purchase intents can be satisfied with less SERPs, which often means a better user experience. We define the daily usage ratio (DUR) as each day's ratio of users who have clicked some values to users whom AliISA has touched. Currently, AliISA serves tens of millions of users per day and achieves a 13.7% DUR, surpassing the panels and the term recommendation system by about 70% and 6% respectively. As an indirect but impactful result, AliISA also introduces a 5% improvement in conversion rate (CVR) with respect to the term recommendation system.

This work makes the following contributions:

- Develops a real-world interactive shopping assistant that earns more usage and conversions from the users.
- Proposes a reinforcement learning (RL) method for tip recommendation where the click-through rate (CTR) of our method leads the baselines by a considerable margin.
- Studies the non-stationary dynamics of an EC environment and proposes some tricks for stabilizing RL.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331409



(a) a non-interactive system. (b) A term recommendation system. (c) AliISA.

Figure 1: Different kinds of shopping assistants.

2 SYSTEM OVERVIEW

We present the architecture of AliISA in Fig. 2. The input layer is the search entrance of our EC platform which receives many kinds of user behaviors. In the second layer, the context manager maintains the context of a search session and coordinates the other modules to make decisions based on this context. In the bottom, a knowledge graph in EC domain and the user daily logs serve as external knowledge and the sources of training data for some of the above modules.

We show the processing flow of AliISA in Fig. 3. Overall, a search session is an interaction between a user and AliISA where the goal of AliISA is to maximize the CTR of recommended properties. We model the problem by Markov decision process (MDP) and propose a RL method to solve it. At each time step t, the context manager maintains a context s_t to represent the current state of this search session. Its components include the user profile (e.g., gender, age, and purchase power), the query q_t, the category c_t, clicked properties/items, recommended properties, etc. The intention identification module is invoked to classify q_t into one category c_t. Then the candidate property generation module provides a set of candidate properties $\mathcal{A}(s_t)$. The property selection module selects either one property from $\mathcal{A}(s_t)$ or a special property $a^{(\text{null})}$ indicating no tip recommendation. At the same time, AliISA acquires a SERP from a traditional item search engine with q_t. If $a_t = a^{(\text{null})}$, this SERP will stay unchanged. Otherwise, the item at a certain position in this SERP will be replaced by the selected property a_t as well as several values that a_t might take. The values are provided by the value ranking module. At the user side, his/her possible behaviors include: (1) click some property values, (2) scroll, (3) make a purchase, (4) exit our service, (5) enter another query. In the case of (1), context manager augments q_{t+1} by clicked values. For (2), $q_{t+1} = q_t$. For both (3) and (4), context manager terminates current search session without preparing a SERP. As for (5), context manager triggers a novel search session.

3 SYSTEM FEATURES

3.1 Intention Identification

There are billions of items in our EC platform, each of which belongs to a certain category $c \in C$. We define an intention as a category and utilizes a gradient boost decision tree (GBDT) model to identify a category from the given query: $c_t = \arg\max_c \Pr(c|q_t)$. For ambiguous queries like "apple" which may refer to either the fruit or the famous company, AliISA emerges only when it is confident enough at its identification. We formalize this idea as a condition: $\max_c \Pr(c|q) \geq T_c$ where T_c is a pre-defined threshold.

3.2 Candidate Property Generation

We have built a knowledge graph in EC domain consisting of more than 100M relations where the entities include categories (C), properties ($\mathcal{A}$), and property values ($\mathcal{V}$), and the predicates include "is-A", "has-property", "has-value", etc. We first acquire $\mathcal{A}(s_t) = \{a|(c_t, \text{has-property}, a)\}$ as the candidate properties from it. Then we need to remove some redundant properties from this set. Basically, properties that have been recommended must be eliminated. We also employ entity linking and slot filling techniques [3] to discover properties that have been explicitly specified by q_t. Taking the query "earphone Beats" for instance, once "earphone" and "Beats" are correctly linked to their corresponding entities, any property entity a (brand in this example) that satisfies (earphone, has-property, a)∧(a, has-value, Beats) should be removed.

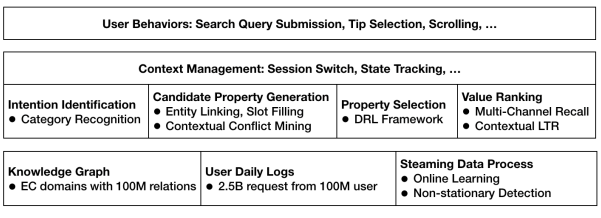


Figure 2: The architecture of AliISA.

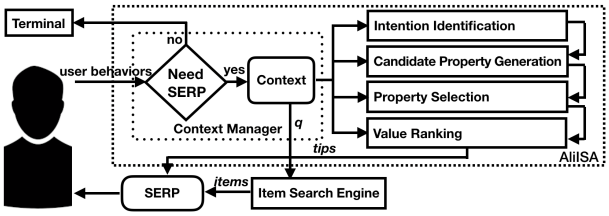


Figure 3: The processing flow of AliISA.

3.3 Property Selection

We first conduct a case study to show the necessity of adopting a RL method. According to the CTRs reported in Fig. 4, a policy being greedy at each SERP will recommend "price range" at the 2nd SERP and "brand" at the 3rd SERP. However, a policy taking long-term considerations will recommend the properties in the reverse order, as this leads to the highest averaged CTR. Thus, we model the property selection task as a MDP and employ deep Q-network (DQN) [8] to solve it.

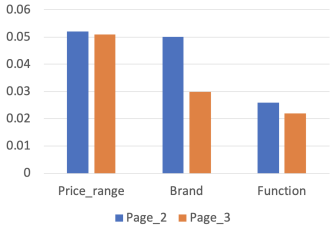


Figure 4: A Case study: greedy leads to sub-optimal.

We parse user daily logs to produce tuples in the form (s_t, a_t, u_t, s_{t+1}) where u_t denotes the user behavior in response to the SERP affected by a_t. To encourage a policy that maximizes the CTR, we define the reward function as: $r_t = 1$ if $a_t \neq a^{\text{null}}$ and u_t is a click operation; $r_t = -\epsilon$ if $a_t \neq a^{\text{null}}$ but u_t is not a click; Otherwise, $r_t = 0$. The ϵ is a pre-defined positive scalar for penalizing tips ignored by our users. The larger ϵ is, the more conservative learnt policy is, say that taking a^{null} more frequently to avoid useless tips. In practice, we take ϵ to be 0.03. Experiments conducted on one of the world's largest EC platform show that our RL method outperforms a SL method by 7% in CTR and 5% in DUR respectively (see Tab. 1).

Method	PV	CTR	DUR
Statistics Ranker	1	1	1
SL	1	1.13	1.08
RL	0.96	**1.21**	**1.13**

Table 1: Comparison between RL method and baselines.

The dynamics of a EC environment are determined by user behaviors, making the environment severely non-stationary. To stabilize RL, we categorize the environmental changes into two kinds and propose solutions respectively.

First, user behaviors change along time within each day due to a roughly common daily routine of the users. We present these changes in Fig. 5 where the PV and CTR of property "brand" change with a surprisingly similar pattern in each of the three days, even though the policy is fixed. We also observe such smooth and periodic changes over many other properties.Thus, denoting the local time of a user as h:m:s, we propose to include the normalized moment $\frac{60h+m}{24\times60}$ as one feature of our state representation s_t.

Figure 5: Periodic changes of the dynamics.

Another kind of environmental changes come from promotion (e.g., "Double 11" day) which often causes fluctuation in the dynamics. Taking Fig. 6 for instance, the CTR of property "brand" drops down drastically within the highlighted range, since there is a promotion of lower-priced items and users participated are less sensitive to the property "brand". A comprehensive study on historical data also suggests that, in most cases, such kinds of fluctuations happen over some user groups.Considering the top-30 head user groups possess more than 70% visits, we monitor the CTR over each of these groups. Once any fluctuation over the CTRs of some groups is detected, we activate the exploration and online learning of DQN over the samples of those groups. In this way, our policy made less recommendation of the property "brand" during the promotion, say that reducing the PV of "brand" by 3%.

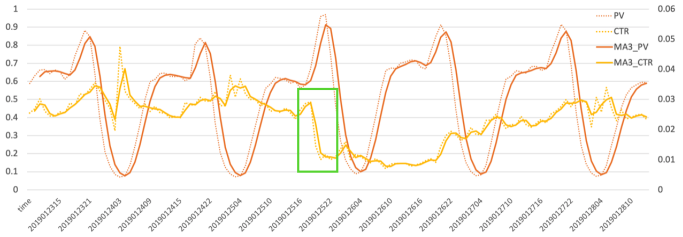


Figure 6: Fluctuation of the dynamics due to promotion.

3.4 Value Ranking

If $a_t \neq a^{\text{null}}$, AliISA needs to provide several values of the property to form a tip. We first recall candidate values from both a query-specific and a personalized channels. The former makes decisions based on the associations between q_t and all possible values. The latter collects items that have been clicked by the user and uses these items to vote for the personally preferred values. Then we apply a GBDT to rank the top-k values for displaying. Meanwhile, we remove any value that would lead to an empty SERP, if it is clicked.

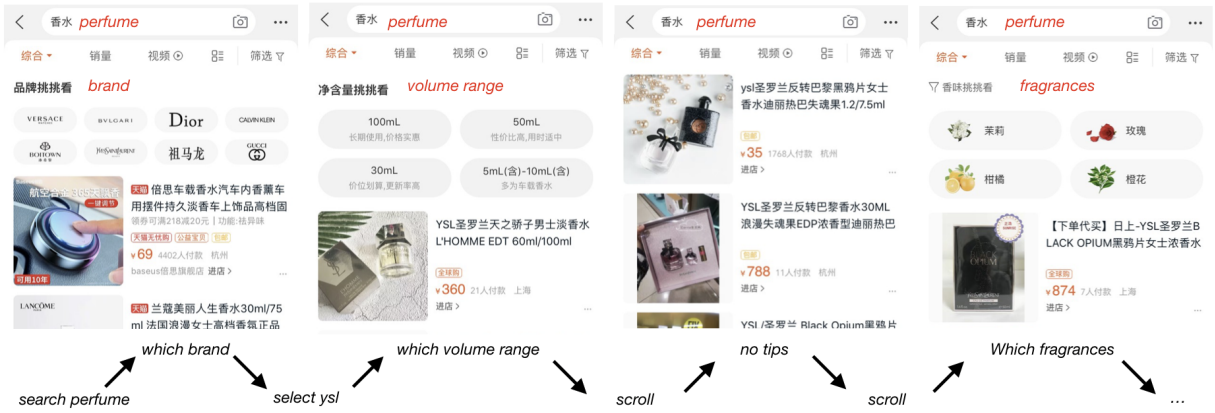


Figure 7: Interactive Search: a typical search session with AliISA's shopping assistance.

4 DEMONSTRATION

We pick a typical search session from user daily logs and present it in Fig. 7, showing how the users interact with AliISA and how we assist our users to quickly reach what they need. Interested readers can access AliISA via Taobao app or watch our demo at YouTube.

At the beginning, the user submits a very general query—"perfume". Our system correctly detects its category as perfume and asks the user for which brand is preferred in the first SERP. The user clicks the value "YSL" and thus, as you can see in the following SERPs, all the highly ranked items are YSL perfumes. Our system offers a tip for the user to specify the volume range of interested perfumes. However, the user ignores this tip and just scroll, revealing that he/she is not sensitive to this property. As a result, the third SERP consists of perfume items in various volume ranges. AliISA provides no tip in the third SERP and receives a scrolling. We must clarify that the user has clicked some items of rose fragrance in this page before scrolling, which cannot be demonstrated due to the limited space. The context manager of AliISA records these click behaviors and uses some components of s_t to represent the history of user behaviors. Our DQN model may notice such a signal and provides the fragrance property in the fourth SERP.

5 RELATED WORK

RL to Rank Traditional learning to rank mainly optimizes the quality of a single SERP [6]. [9] show that commonly used metrics for measuring ranking quality can be easily expressed by a cumulative reward, making RL to rank promising. The ultimate goal of EC search engines is usually to improve the transaction amount [5]. The most related work to this paper is tip recommendation [1] which also maximizes the CTR of displayed tips. However, the tip in their design is a collection of related terms (see Fig. 1b).

Non-stationary Dynamics Most RL algorithms [8] are designed for static environments. [2] introduce RL with context detection which provides an unified framework for studying the non-stationary issues. However, most approaches in this line [4] are designed for tabular cases, limiting their usage in our system. [1] propose to use stratified sampling and a baseline for better reward estimation. Their techniques have been confirmed effective in stabilizing the learning procedure of DDPG [7] under an EC environment.

6 CONCLUSION

In this paper, we present AliISA which embeds a certain property and its values into the SERP for users to further specify the query. We propose a RL method for property selection with some tricks to handle the non-stationary issue. Currently, our system serves tens of millions of users per day, and more than 10% of them have benefited from our assistance. Compared with existing systems, our system introduces a 5% improvement in CVR, which means a remarkable commercial impact. We plan to seek the optimal policies for both property selection and value ranking via hierarchical RL.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Research and Development Program of China (2018YFB1003405), the National Natural Science Foundation of China (61702286), the Natural Science Foundation of Tianjin, China (18JCYBJC15600). Xiaoli Gong is corresponding author.

REFERENCES

[1] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *SIGKDD'18*.
[2] Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. 2006. Dealing with non-stationary environments using context detection. In *ICML'06*.
[3] Yu Gong, Xusheng Luo, Yu Zhu, Wenwu Ou, Zhao Li, Muhua Zhu, Kenny Q Zhu, and Xi Chen Lu Duan. 2019. Deep Cascade Multi-task Learning for Slot Filling in Online Shopping Assistant. In *AAAI'19*.
[4] Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. 2014. Sequential decision-making under non-stationary environments via sequential change-point detection. In *LMCE'14*.
[5] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *SIGKDD'18*.
[6] Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94, 10 (2011).
[7] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. In *ICLR'16*.
[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A Riedmiller. 2018. Playing Atari with Deep Reinforcement Learning. In *NIPS'18*.
[9] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *SIGIR'17*.