Query-based Interactive Recommendation by Meta-Path and Adapted Attention-GRU

Yu Zhu*
Alibaba Group
Hangzhou, China
zy143829@alibaba-inc.com

Yu Gong*
Alibaba Group
Hangzhou, China
gongyu.gy@alibaba-inc.com

Qingwen Liu
Alibaba Group
Hangzhou, China
xiangsheng.lqw@alibaba-inc.com

Yingcai Ma
Alibaba Group
Hangzhou, China
yingcai.myc@alibaba-inc.com

Wenwu Ou
Alibaba Group
Hangzhou, China
santong.oww@taobao.com

Junxiong Zhu
Alibaba Group
Hangzhou, China
xike.zjx@taobao.com

Beidou Wang
State Key Laboratory of CAD&CG,
Zhejiang University
Hangzhou, China
beidou.wang@gmail.com

Ziyu Guan
Xidian University
Xi'an, China
zyguan@xidian.edu.cn

Deng Cai
State Key Laboratory of CAD&CG,
Zhejiang University
Hangzhou, China
dcai@zju.edu.cn

ABSTRACT

Recently, interactive recommender systems are becoming increasingly popular. The insight is that, with the interaction between users and the system, (1) users can actively intervene the recommendation results rather than passively receive them, and (2) the system learns more about users so as to provide better recommendation.

We focus on the single-round interaction, i.e. the system asks the user a question (Step 1), and exploits his feedback to generate better recommendation (Step 2). A novel query-based interactive recommender system is proposed in this paper, where **personalized questions are accurately generated from millions of automatically constructed questions** in Step 1, and **the recommendation is ensured to be closely-related to users' feedback** in Step 2. We achieve this by transforming Step 1 into a query recommendation task and Step 2 into a retrieval task. The former task is our key challenge. We firstly propose a model based on Meta-Path to efficiently retrieve hundreds of query candidates from the large query pool. Then an adapted Attention-GRU model is developed to effectively rank these candidates for recommendation. Offline and online experiments on Taobao, a large-scale e-commerce platform in China, verify the effectiveness of our interactive system. The system has already gone into production in the homepage of Taobao App since Nov. 11, 2018 (see https://youtu.be/hAkXDOf2dDU on how it works online). Our code is public in https://github.com/zyody/QueryQR.

*Both authors contributed equally to this research.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**; • **Human-centered computing** → **Social recommendation**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Recommender Systems, Meta-Path, Attention-GRU, Wide&Deep Learning

ACM Reference Format:
Yu Zhu, Yu Gong, Qingwen Liu, Yingcai Ma, Wenwu Ou, Junxiong Zhu, Beidou Wang, Ziyu Guan, and Deng Cai. 2019. Query-based Interactive Recommendation by Meta-Path and Adapted Attention-GRU. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357805

1 INTRODUCTION

Interactivity plays an important role in influencing user experience in daily life. For example, compared to watching TV, most children prefer playing with the smartphone or tablet. One main reason is that, children can only passively receive TV shows from TV (with few interactions when they change channels), while they have lots of interactions with the smartphone or tablet when, for example, playing mobile games.

It is the same case with recommender systems (RS). Intuitively, by introducing the interactivity in RS (i.e. users interact with RS), users can actively intervene the recommendation results rather than passively receive them. In addition, the system will learn more about users so as to provide better recommendation. Both will improve the user experience. However, how to utilize the interactivity to improve the recommendation performance has not been well studied in the past decades.

There is an increasing number of works [2, 6] concentrating on interactive RS recently. Personal assistants learn users' preferences by conversing with users [15]. Users' interests can also be mined

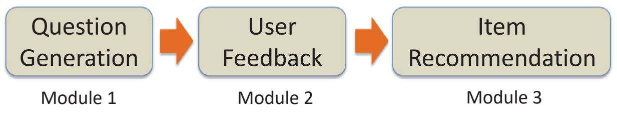


Figure 1: The Question&Recommendation framework.

by questionnaires [8]. Reinforcement learning [2] /multi-armed bandit [17]/active learning [22] methods focus on balancing the explore-exploit tradeoff in RS. They learn users' preferences by recommending explored items and acquiring their feedback. However, personal assistants and reinforcement learning/multi-armed bandit/active learning methods usually need multi-round interactions to well learn users, thus users who want quick&accurate recommendations would not be satisfied. The questionnaires are often manually generated and not well personalized. These issues prevent the above methods from being the best fit for interactive recommendation tasks.

Single-round interaction and automatically generating personalized questions are promising solutions to the aforementioned challenges. A framework for the single-round interactive RS is proposed in [6], which contains three main modules as shown in Figure 1. Specifically, when a user browses items in RS, the system would generate a question to consult him about what his interest is (Module 1). Once his feedback is obtained (Module 2), the system could then provide more accurate recommendation (Module 3).

Our work is also based on this framework. Now we describe our design in terms of the three modules, respectively.

***Question Generation*:** We transform the question generation task into a query recommendation task. For example, in Figure 2 (a), four queries "Hat", "Scarf", "Glove" and "Socks" are recommended to the user, corresponding to the questions "Do you want to buy a hat/scarf/glove/socks?". We choose to generate questions based on queries due to the following three considerations:

- Queries are typed by users, in order to find their preferred items. Therefore, queries reflect users' potential preferences.
- The questions generated in [6] are topic-based. Compared to topics, queries can capture more fine-grained preferences.
- The search log has plenty of queries. These queries are widely distributed and cover almost all preferences from different users in various circumstances (e.g. different seasons).

After filtering queries with low frequency, millions of queries are collected from the search log. To recommend queries (from the large query pool) that can best reflect users' preferences in an efficient fashion, various important information in e-commerce websites should be well exploited, including (1) heterogeneous relations among different objects (users, items, queries, etc.); (2) rich features of these objects, e.g. the text and category (Dresses, Smartphones, etc.) information of items and queries; (3) the sequential information, action types (*click*, *purchase*, etc.) and timestamps of users' behaviors. Inspired by [7], we split query recommendation into two stages: Candidate Generation and Ranking. Specifically, we firstly propose a model based on Meta-Path [30] to efficiently generate hundreds of query candidates from the large query pool, with the help of heterogeneous relations. Then an adapted Attention-GRU

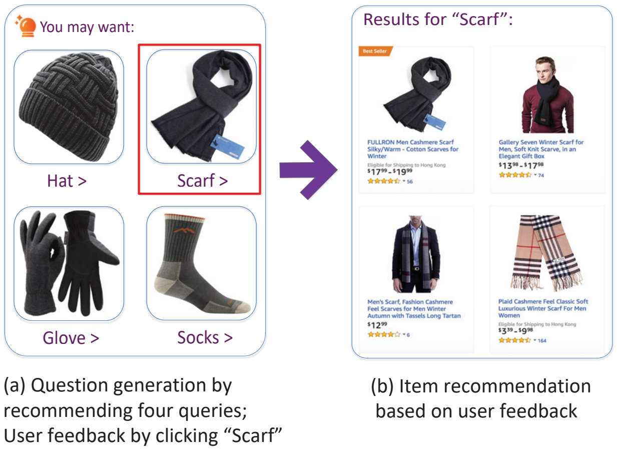

You may want:
Hat >
Scarf >
Glove >
Socks >
Results for "Scarf":
(a) Question generation by recommending four queries; User feedback by clicking "Scarf"
(b) Item recommendation based on user feedback

Figure 2: The system generates 4 queries to consult the user about whether he wants a hat/scarf/glove/socks. He answers "Yes, I want a scarf" by clicking "Scarf". Then personalized item recommendation is performed based on the query "Scarf".

[5] model is developed to effectively rank these candidates, by utilizing all the information described above. In this way, high efficiency and accurate recommendation are both achieved.

***User Feedback*:** As shown in Figure 2 (a), the user answers "Yes, I want a scarf" by clicking "Scarf" or "No, I do not want them" by ignoring them.

***Item Recommendation*:** Finally, as shown in Figure 2 (b), items are recommended based on the clicked query and the user's historical behavior. Actually, this can be seen as a classical personalized retrieval task, and many successful algorithms in the information retrieval area could be adopted. In this way, the recommendation (i.e. the retrieved results) is ensured to be closely-related to users' feedback (i.e. query).

Apart from the motivated example in Figure 2, we show some other possible user cases to give a better understanding of our system:

- After the user clicking some items about trips to Tokyo, he is recommended with queries "Hotel in Tokyo", "Flight to Tokyo", "Travelling bag" and "Toiletries".
- After buying a Nikon camera, he is shown with "Memory card", "Lenses", "Camera battery" and "Protection cases for camera".
- After favoring a pair of shoes, he is shown with "Nike shoes", "Adidas shoes", "Sneakers" and "Leather shoes".

Since item retrieval algorithms in most e-commerce platforms are mature, we directly adopt them as the models in *Item Recommendation*. Therefore, query recommendation in *Question Generation* is our key challenge. We will focus on it in the rest of our paper.

This paper's contributions are outlined as follows.

- We design a novel query-based interactive RS. Compared to state-of-the-art interactive RS, e.g. [6], our system can accurately generate personalized questions from millions of automatically constructed questions (since we have millions of queries) and item recommendation is ensured to be closely-related to users' feedback, which result in better user experience in interactive RS.

- We propose a solution by Meta-Path and adapted Attention-GRU for query recommendation. This solution follows a Candidate Generation and Ranking schema. We introduce Meta-Path into the Candidate Generation stage and customize the calculation of meta path scores, so that query candidates can be efficiently generated considering heterogeneous relations and the procedure is more explainable. We introduce Attention-GRU into the Ranking stage and propose two important modifications on Attention-GRU, which significantly improves its ranking performance.
- We conduct extensive offline and online experiments on a large-scale e-commerce platform, i.e. Taobao. The experimental results (especially the response from online users) prove the effectiveness of our query-based interactive RS.

2 RELATED WORK

2.1 Interactive RS and Query Recommendation

[10, 15] are excellent surveys on interactive RS. How we differ from the most relevant works are described in Introduction. In addition, compared to [6], our solution in *Question Generation* is based on queries and is carefully designed to improve its efficiency and effectiveness. Meanwhile, we transform *Item Recommendation* into a retrieval task so that we can address it by adopting existing retrieval algorithms.

Query recommendation in most previous works [14, 31] is to facilitate the search of web pages, locations, etc. They usually exploit information, e.g. searched queries and clicked links, in search logs. Ours is for item recommendation, and is based on both of search and recommendation logs, including some special information in e-commerce websites. Note that our framework is not limited to queries and other objects (e.g. videos) can also be utilized to generate questions. *Item recommendation* will then be based on behaviors on these objects, instead of only the clicked queries. We will explore it in our future work.

2.2 Meta-Path

Meta-Path [24] describes how two nodes in a graph are connected via different types of paths. Specifically, given a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{V_0, V_1, \cdots\}$ is the node set, and $\mathbf{E} = \{E_0, E_1, \cdots\}$ is the edge set. A meta path $P = V_0 \xrightarrow{E_0} V_1 \xrightarrow{E_1} \cdots \xrightarrow{E_{k-1}} V_k$ in $\mathbf{G}$ defines a complicated relation between V_0 and V_k. Several works [28, 29] exploit Meta-Path to improve the performance of RS. Corresponding to RS, entities such as users and items construct the nodes, and relations such as users consuming items are the edges. Many recommendation algorithms can be represented by meta paths. For example, item-based collaborative filtering (Item-CF) [23] and Content-based recommendation (CBR) [20] can be represented by meta paths: $p^{Item-CF}/p^{CBR} = u \xrightarrow{Consume} i \xrightarrow{Similar} i'$, indicating that user u consuming item i may also prefer a similar item i' (the similarity is calculated by collaborative behaviors for Item-CF and by item contents/attributes for CBR). Similarly, user-based collaborative filtering (User-CF) [32] and Social-aware recommendation (SR) [25] are represented by: $p^{User-CF}/p^{SR} = u \xrightarrow{Similar} u' \xrightarrow{Consume} i$, indicating that u may favor what his similar users (the similarity is calculated by collaborative behaviors for User-CF and by social relations for SR) have consumed. [30] proposes a Meta-Graph based recommendation method to capture more complex semantics. Since there is no complex semantics in our task, thus we use Meta-Path for simplicity. Compared to previous Meta-Path models, we combine Meta-Path with Attention-GRU, instead of using Meta-Path individually. Thus heterogeneous relations and sequential behaviors are both considered for recommendation.

2.3 Attention-GRU

Attention-GRU [5] refers to GRU [4] with the attention schema [1]. It typically generates an output sequence $y = (y_1, \cdots, y_T)$ from an input sequence $x = (x_1, \cdots, x_n)$, where x is usually encoded to a sequential representation $h = (h_1, \cdots, h_n)$ by an *encoder*. y_m in y is generated by:

$$\alpha_m = Attend(s_{m-1}, h), \tag{1}$$

$$g_m = \sum_{k=1}^{n} \alpha_{mk} h_k, \tag{2}$$

$$s_m = Recurrence(y_{m-1}, s_{m-1}, g_m), \tag{3}$$

$$y_m \sim Generate(y_{m-1}, s_m, g_m), \tag{4}$$

where *Attend* and *Generate* are functions. s_m is the hidden state. α_m is a vector whose entry α_{mk} indicates the *attention weight* of the k-th input. g_m is called a *glimpse* [19]. *Recurrence* represents the recurrent activation. In Attention-GRU, the recurrent activation is GRU.

RNN solutions for behavior modeling are becoming increasingly popular [11, 12]. The most related work to ours is [36]. Our main difference lies in (1) we modify the attention schema in Attention-GRU while [36] does not; (2) Besides behavior features extracted by Attention-GRU, we also incorporate other valuable features (e.g. features from Meta-Path) for recommendation in a non-trivial fashion.

3 QUERY RECOMMENDATION

As described in Introduction, query recommendation is our key challenge. Our solution contains two stages: Candidate Generation and Ranking.

3.1 Meta-Path for Candidate Generation

Candidate Generation is to efficiently generate hundreds of query candidates from the large query pool. As shown in Figure 3, we design three types of meta paths to generate candidates: U2I2Q, U2I2S2Q and U2I2C2Q.

U2I2Q: Based on search logs in previous days, we calculate the conditional probability of query q given item i as follows:

$$P(q/i) = \frac{Count(q, i)}{Count(i)}, \tag{5}$$

where $Count(q, i)$ is the number of records that the retrieved items contain i by searching q, and $Count(i)$ is the number of all records

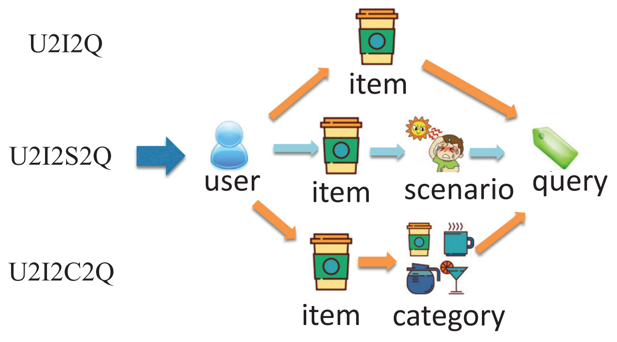


Figure 3: Three types of meta paths to generate query candidates.

that i is retrieved. The insight is that, if q and i always co-occur in search logs, then they are closely related. For a given user u, it indicates that u could find his preferred queries by the meta path: $p^{U2I2Q} = u \xrightarrow{Consume} i \xrightarrow{P(q/i)} q$.

***U2I2S2Q*:** U2I2S2Q contains two sub-procedures: I2S and S2Q. I2S indicates that when user u consumes item i, it would activate a scenario s, with $P(s/i)$ calculated considering i's title and category. For S2Q, we derive the conditional probability of other items i' given s, i.e. $P(i'/s)$, based on i''s titles and categories. We then obtain $P(q/s)$ with the help of $P(q/i')$ in Eq. (5). Finally, u could find his preferred queries by the meta path: $p^{U2I2S2Q} = u \xrightarrow{Consume} i \xrightarrow{P(s/i)} s \xrightarrow{P(q/s)} q$.

***U2I2C2Q*:** In order to generate more query candidates, we further construct the meta path: $p^{U2I2C2Q} = u \xrightarrow{Consume} i \xrightarrow{P(c/i)} c \xrightarrow{P(q/c)} q$, where $P(c/i) = 1$ if item i belongs to category c. $P(q/c)$ is calculated based on the knowledge graph.

We omit the description on how to calculate $P(s/i)$, $P(q/s)$ and $P(q/c)$ in detail since it is not the focus of this paper. $MetaScore(q)$ denotes the relation weight between u and q in terms of a certain type of meta path, defined as:

$$MetaScore(q) = \begin{cases} \sum_{p \in U2I2Q} P(q/i) & \text{if } q \text{ is generated by U2I2Q} \\ \sum_{p \in U2I2S2Q} P(s/i) \times P(q/s) & \text{if } q \text{ is generated by U2I2S2Q} \\ \sum_{p \in U2I2C2Q} P(c/i) \times P(q/c) & \text{if } q \text{ is generated by U2I2C2Q} \end{cases} . \quad (6)$$

We collect u's recently consumed (*clicked, purchased, favored, added-to-cart*) items as i. The weight of a meta path is the product of different conditional probabilities along the meta path. The probability product is used because, we assume that in Meta-Path, the observation on one node only depends on its previous node, i.e. following the first-order Markov assumption. Taking a certain meta path in U2I2S2Q as an example, its probability of occurrence $P(i, s, q)$ (item i, scenario s and query q are along the meta path) is equal to $P(i) \times P(s/i) \times P(q/s)$ according to the Markov assumption. $P(i)$ is assumed to be the same for all items recently consumed by the target user. Therefore, $P(s/i) \times P(q/s)$ in Eq. (6) is proportional to $P(i, s, q)$. The other types of meta paths can be similarly analyzed. There may exist multiple meta paths that link u and q, thus we sum their weights to obtain $MetaScore(q)$.

For a query candidate, we represent the output of Meta-Path by a 6-dimensional vector: $[Type_1, Score_1, Type_2, Score_2, Type_3, Score_3]$. If there exists at least one meta path belonging to U2I2Q that links u and q, then we have $Type_1 = 1$ and $Score_1$ is $MetaScore(q)$ with respect to U2I2Q. Otherwise, we have $Type_1 = 0$ and $Score_1 = 0$. $Type_2$, $Score_2$ are for U2I2S2Q and $Type_3$, $Score_3$ are for U2I2C2Q.

We calculate $i \to \cdots \to q$ relations, save i's top-k queries with the largest conditional probabilities and build index on i for each type of meta path offline. Then in online recommendation, given user u, query candidates can be efficiently generated according to u's consumed items by indexing. In our system, we set the maximum number of query candidates for each type of meta path to be 200. In this way, the total number of query candidates for each user request is controlled to be less than or equal to 600 (since we have three types of meta paths). Note that other relations apart from the above three can also be incorporated in our Meta-Path model. Compared to the embedding model for Candidate Generation in [7], ours exploits heterogeneous relations to generate candidates, is more explainable and is easier to implement.

3.2 Attention-GRU Based Model for Ranking

The Ranking stage is to effectively rank query candidates and top queries are then recommended. We formulate it as a point-wise ranking task. Specifically, a classifier is learned by exploiting various information in e-commerce websites. Then given user u, his probability of preferring each query is predicted by the classifier and queries are ranked by their probabilities. As shown in Figure 4, there are 3 feature fields in our ranking model: user features, query features and context features.

***user features*:** This feature field contains users' discrete features (e.g. user id), continuous features (e.g. age) and behavior features. We encode discrete features by embedding. Taking user id as an example, the user set is denoted by $U = \{u_1, u_2, \cdots, u_N\}$. u_k's one-hot vector is defined as $o_k \in \{0, 1\}^{N \times 1}$, with the k-th entry equal to 1 and the other entries equal to 0. Then we obtain u_k's embedding e_k as:

$$e_k = \mathbf{E} \times o_k. \quad (7)$$

$\mathbf{E} \in R^{D \times N}$ contains the embeddings of all users, which is learned from training. o_k is used to look up the embedding of u_k from $\mathbf{E}$. Behavior features are represented by the hidden state of our modified Attention-GRU, considering the text information, categories, other discrete and continuous features of users' consumed items. The representation of text is not the focus of this paper, thus we simply represent it by the mean of word embeddings. Categories are discrete features, thus they are directly encoded by embedding.

***query features*:** This feature field contains the text information, categories, Meta-Path features, other discrete and continuous features of the query. Note that there is no explicit category information on queries, thus we learn a model to predict the top 3 categories for each query, and encode them by the mean of category embeddings.

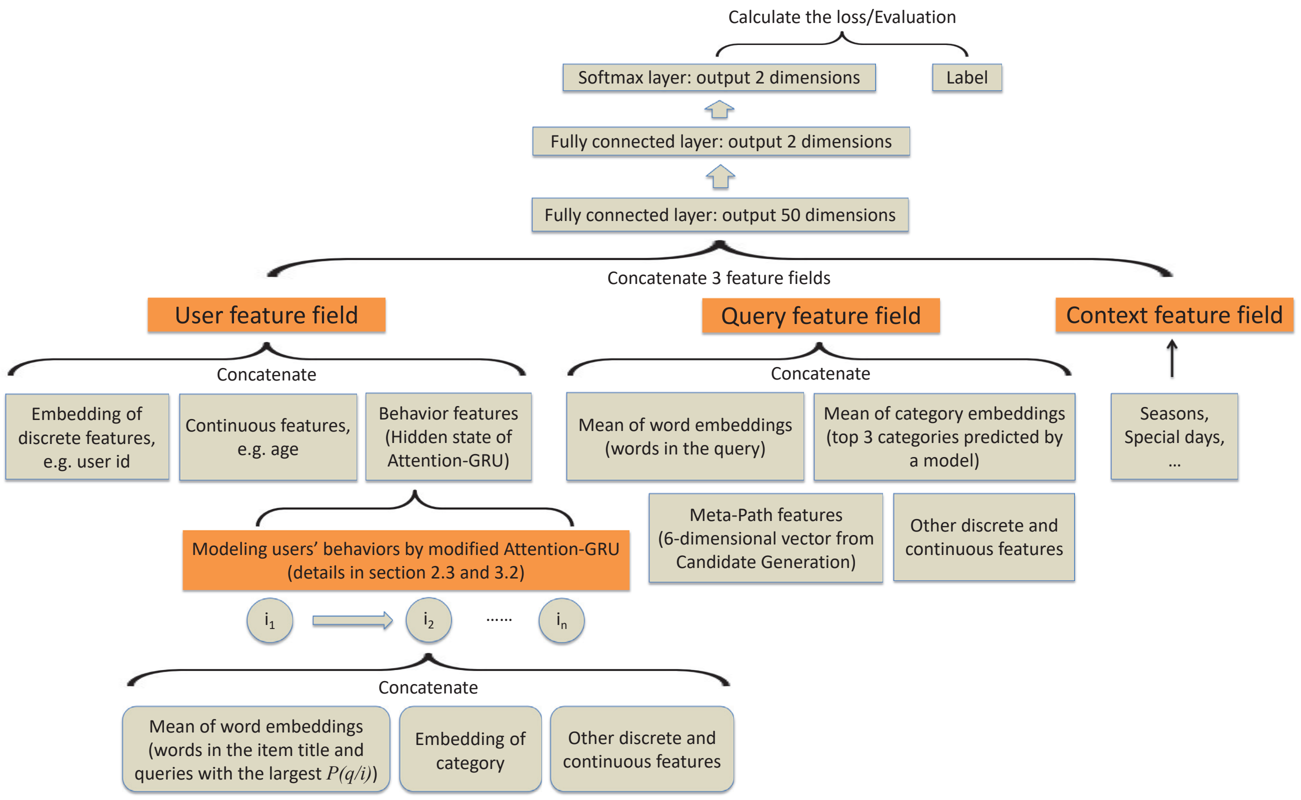

Calculate the loss/Evaluation
Softmax layer: output 2 dimensions
Label
Fully connected layer: output 2 dimensions
Fully connected layer: output 50 dimensions
Concatenate 3 feature fields
User feature field
Query feature field
Context feature field
Concatenate
Embedding of discrete features, e.g. user id
Continuous features, e.g. age
Behavior features (Hidden state of Attention-GRU)
Concatenate
Mean of word embeddings (words in the query)
Mean of category embeddings (top 3 categories predicted by a model)
Seasons, Special days, ...
Modeling users' behaviors by modified Attention-GRU (details in section 2.3 and 3.2)
Meta-Path features (6-dimensional vector from Candidate Generation)
Other discrete and continuous features
i1
i2
......
in
Concatenate
Mean of word embeddings (words in the item title and queries with the largest P(q/i))
Embedding of category
Other discrete and continuous features

Figure 4: Attention-GRU based model to rank query candidates.

Meta-Path features are the 6-dimensional vector obtained from Candidate Generation.

***context features*:** This feature field contains seasons, special days, etc., which also influence the ranking performance.

The concatenation of these feature fields are the input of a 3-layer neural network (2 fully connected layers and 1 softmax layer). Loss/evaluation is calculated based on its output and the label. Some features capture the system bias, e.g. for the feature of special days, users are more likely to click queries in Shopping Festivals than in normal days. Some features capture the bias on users or queries, e.g. for the feature of user id, some active users prefer to click most queries. Similarly, queries with popular categories tend to be clicked by most users. Other features capture users' personalized preferences on queries, e.g. the text information of a user's consumed items models his preference by text. Then if the query also contains similar text, he would probably prefer this query. Different from linear models, neural network used in our framework not only captures the bias of unary features (i.e. the bias on system, users and items), but also well models the interaction among features (e.g. users' personalized preferences on queries) by non-linear activation functions.

Our ranking model is inspired by the Wide&Deep model [3]. Obtaining behaviour features from Attention-GRU in Figure 4 can be regarded as the deep component in [3] while the other features construct the wide component. However, our model and the Wide&Deep model in [3] share some key differences. For the deep component, we use the hidden state of our modified Attention-GRU as features while [3] regards the final score of Deep Neural Network (DNN) [13] as the feature. Thus our model exploits users' sequential behaviors for recommendation while [3] cannot. For the combination of the wide and deep components, [3] combines them by logistic regression while we use a 3-layer neural network to model them, so that the interaction among features in wide and deep components can be better captured.

Modified Attention-GRU

Now we describe our modified Attention-GRU in detail. Following the denotations in section 2.3, x_k corresponds to item i_k in Figure 4. T is equal to 1, thus we have $m = 1$ in Eq. (1) ~ (4). y_0 corresponds to the query. s_0 is the hidden state of y_0. We use bidirectional GRU to output $h = (h_1, \cdots, h_n)$, with h_k defined as:

$$\overrightarrow{h_k} = GRU(x_k, \overrightarrow{h_{k-1}}), \quad (8)$$

$$\overleftarrow{h_k} = GRU(x_k, \overleftarrow{h_{k+1}}), \quad (9)$$

$$h_k = [\overrightarrow{h_k}, \overleftarrow{h_k}]. \quad (10)$$

The concatenation of h_k (containing the information of i_k) and s_0 (containing the query information) is the input of a neural network, and its output is the attention weight α_{1k}. Finally, we use s_1 in Eq. (3) to represent behavior features in Figure 4. Our modifications on Attention-GRU are based on the following two motivations:

- Different action types (*click*, *purchase*, *favor*, *add-to-cart*) reflect users' different preferences on items, e.g. generally a user purchasing an item indicates he is more interested in the item than if he clicks it. The design of attention weight should consider different action types.
- The earlier an action happens, the less it affects query recommendation. Thus the time decay of different actions should also be modeled in attention weight.

Following these motivations, as shown in Figure 5, we replace h in Eq. (1) with $h' = (h'_1, \cdots, h'_n)$, where h'_k is defined as:

$$h_k^{tmp} = \mathrm{A}_l \times h_k, \tag{11}$$

$$h'_k = h_k^{tmp} \otimes \triangle t_k^{\epsilon}, \; s.t. \; \epsilon \leq 0. \tag{12}$$

Suppose we have $h_k \in R^{d\times 1}$, then we define $\mathrm{A}_l \in R^{d\times d}$, where l indicates whether i_k is *clicked* ($l = 1$), *purchased* ($l = 2$) , *favored* ($l = 3$) or *added-to-cart* ($l = 4$). We use matrix multiplication to model different action types in Eq. (11), since in this way the interaction between the action type and i_k is explicitly captured. $\triangle t_k$ is the time interval between the time when the action on i_k happens and the time for query recommendation. ϵ is an exponential decay on $\triangle t_k$. $\otimes$ indicates that each entry in h_k^{tmp} is multiplied by $\triangle t_k^{\epsilon}$. Constraint $\epsilon \leq 0$ in Eq. (12) ensures that the earlier an action on i_k happens, the smaller h'_k will be. Correspondingly, i_k's influence on query recommendation will also be smaller. A_l and ϵ are learned from training. Constraint $\epsilon \leq 0$ is handled by using the projection operator [21], i.e. if we have $\epsilon > 0$ during training iterations, we reset $\epsilon = 0$. Note that our two modifications can be generalized to other attention models, by replacing their h_k with our h'_k as shown in Eq. (11) and Eq. (12).

4 OFFLINE EXPERIMENTS

4.1 Dataset

We firstly deploy our interactive RS in Taobao, using a heuristic ranking method (we cannot use our ranking model since no training data is available yet) to recommend queries. Then one day of data is collected and preprocessed for training and testing. For each instance <*uid, qid, label*>, *label* = 1 indicates that query *qid* is clicked by user *uid* and *label* = 0 means that *qid* is shown to but not clicked by *uid*. In addition, user features (behavior features are constructed by *uid*'s recent 100 actions), query features and context features are correspondingly collected. Finally, we have 3, 201, 231 users, 1, 464, 410 queries and 12, 897, 055 instances. 80% instances are randomly selected for training and the remaining 20% are for testing.

4.2 Compared Models and Evaluations

Our ranking model is compared with the following baselines.

Q&R: We denote the question ranking model in state-of-the-art interactive RS [6] (replace its topics with queries) as Q&R.

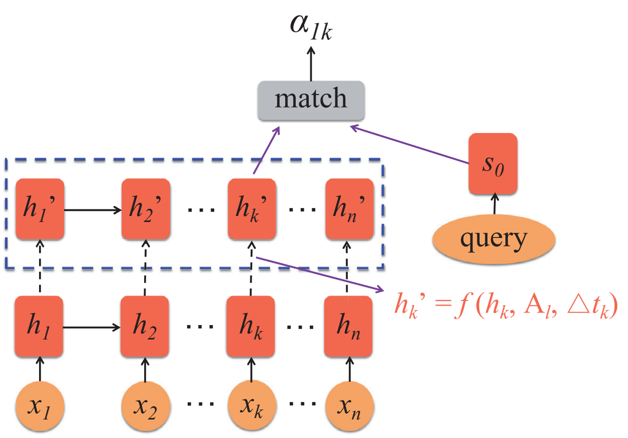


Figure 5: Modified attention schema considering action types and time decay.

FTRL: FTRL [18] is a linear model, with no feature interaction. Behavior features in Figure 4 are not used since they cannot be easily incorporated in FTRL. We manually design some interactive features and behavior features for FTRL to ensure more fair comparisons.

Wide&Deep: Wide&Deep learning [3] is a popular learning framework in industry. Here we compare our model with its famous version described in [3]. Based on the discrete and continuous features in Figure 4, we construct raw input features and transformed features for the wide component, and use DNN to generate real-valued vectors for the deep component. Finally, the wide and deep components are combined by a logistic regression model. Refer to [3] for more details.

The remaining baselines replace our modified Attention-GRU with other RNN structures. We use the corresponding RNN structures to denote these baselines for simplicity.

GRU: GRU [4] is one of the best RNN architectures. Thus it is selected to represent the original RNN structures, with timestamps and action types not considered.

Attention-GRU: Similarly, we choose Attention-GRU [5] to represent RNN structures with the attention schema. Timestamps and action types are not considered, either.

Time-LSTM: Time-LSTM [34] has achieved state-of-the-art performance for sequential behavior modeling when timestamps exist while action types are not known. We use its publicly available python implementation[1].

Attention-GRU-3M: Attention-GRU-3M [36] considers timestamps and action types in sequential behavior modeling. However, its time intervals are calculated between neighbor actions, which are different from ours. In addition, we propose two important modifications on the attention schema to better model timestamps and action types. We use its publicly available python implementation[2].

[1]https://github.com/DarryO/time_lstm
[2]https://github.com/zyody/Attention-GRU-3M

	50% training data		100% training data	
	AUC	F1	AUC	F1
Q&R	0.651	0.638	0.671	0.648
FTRL	0.647	0.635	0.669	0.644
Wide&Deep	0.651	0.639	0.673	0.650
GRU	0.655	0.642	0.676	0.653
Attention-GRU	0.661	0.649	0.683	0.662
Time-LSTM	0.664	0.651	0.684	0.666
Attention-GRU-3M	0.671	0.657	0.690	0.674
Our Model	**0.682***	**0.667***	**0.699***	**0.685***

Table 1: Model Comparison (* indicates statistical significance at $p < 0.01$ compared to the second best.)

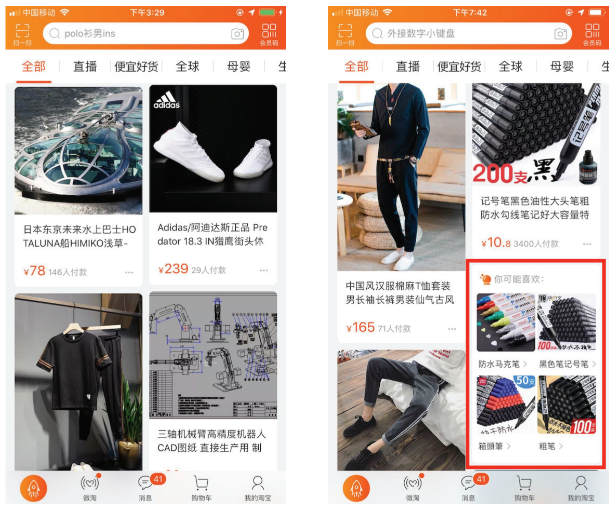
	AUC	Gain
Random prediction	0.500	-
Add categories of the query	0.568	+0.068
Add categories of sequential items	0.604	+0.036
Add texts of sequential items and the query	0.620	+0.016
Add all features in Figure 4	0.683	+0.063
Add the modification in Eq. (11)	0.689	+0.006
Add the modification in Eq. (12)	0.699	+0.010

Table 2: Evaluation on Different Ingredients of Our Ranking Model (*Gain* represents the improvement of current AUC compared to the previous one.)

The number of units is set to 256 for all RNN-based structures. The other hyper-parameters in all models are tuned via cross-validation or set as in the original paper. We evaluate the performance of different models by AUC and F1 score [16].

4.3 Results and Discussions

4.3.1 Model Comparison. As shown in Table 1, our proposed model significantly outperforms all baselines. In comparison, Q&R predicts the query conditioned on a sequence of consumed items, with features not carefully engineered. Moreover, it uses GRU, which is inferior compared to our modified Attention-GRU. FTRL is a linear model, with interactive features and behavior features manually designed. However, it is difficult for FTRL to capture complex feature interactions and well model sequential behaviors. The DNN used in Wide&Deep fails to model sequential behaviors, either. In addition, its combination model, i.e. logistic regression, is linear, which cannot well capture the interaction among features in the wide and deep components. Attention-GRU performs better than GRU due to the attention schema, but it is worse than our model, which demonstrates that adding timestamps and action types in behavior modeling can improve the ranking performance. Time-LSTM fails to distinguish different action types. Attention-GRU-3M considers timestamps and action types, but still performs worse than our model, which proves the advantage on how we model timestamps and action types.



(a) The original recommendation setting.

(b) Occasionally presenting the user interface of our interactive RS, highlighted by the red rectangle.

Figure 6: A/B test setting. (a) is the original setting. Our interactive RS is added in (b).

4.3.2 Contributions of Different Ingredients. We further conduct experiments to evaluate the contributions of different ingredients in our ranking model. As shown in Table 2, random prediction achieves an AUC of 0.500. If we only use the query's categories as input, AUC reaches 0.568, which indicates the popularity of query's categories is useful for query recommendation. When the categories of sequential items are added, AUC increases to 0.604, demonstrating that item categories and their sequential information contribute to performance improvement. Adding the text information of sequential items and the query further increases AUC to 0.620, which verifies the effectiveness of texts. Attention-GRU with all features in Figure 4 achieves an AUC of 0.683, proving the usefulness of other features. Finally, our modifications on attention schema, i.e. Eq. (11) and Eq. (12), obtain the AUC gain of 0.006 and 0.010, respectively, which verifies the effectiveness of our two modifications.

5 ONLINE EXPERIMENTS

We now test real users' response to our interactive RS in Taobao. In this large-scale platform, there are over 10^{10} *impressions* and about 6×10^8 *clicks* on over 10^7 items from nearly 7×10^7 customers within one normal day. A standard A/B test is conducted online, where one adopts the original recommendation setting as shown in Figure 6 (a) and the other occasionally presents the user interface of our interactive RS as shown in Figure 6 (b). The appearance of interactive user interfaces is controlled by an intention model, and often happens after the user clicks some items and returns back to the homepage of Taobao App. See https://youtu.be/hAkXDOf2dDU on how it works online. The same number (about 7×10^6 per day) of users are randomly selected for each setting. We perform the online experiments for five days, and the average impression number on

	Impression	Click	GMV (CNY)
Original setting	1448730074	62956757	9261422
Add our interactive RS	1469132334	63361783	9532320
Improvement	1.41%	0.64%	2.93%

Table 3: Results of Online Experiments (*Improvement* is a relative growth of *Add our interactive RS* compared to *Original setting*, e.g. $2.93\% \approx (9532320 - 9261422)/9261422$**.)**

items (denoted as *Impression*), click number on items (denoted as *Click*) and Gross Merchandise Volume (denoted as *GMV*) per day are reported.

As shown in Table 3, by adding our interactive RS, *Impression*, *Click* and *GMV* are all improved. A higher *Impression* and *Click* indicates that users are more willing to browse and click items in our interactive RS. The improvement of *GMV* is larger, because based on users' feedback, the system can well learn users' shopping needs and then satisfy them, which would lead to much more purchases. Considering the platform's traffic, 2.93% improvement on *GMV* would result in a significant boost in revenue.

In our platform, if we increase the number of query candidates generated by the Candidate Generation stage, both of users' Click-Through Rate (denoted as *CTR*) on queries and the *response time* will increase. When the number is larger than 900, it will exceed the system's constraint on *response time*. *CTR* increases because the model in the Ranking stage is more effective than the one in the Candidate Generation stage. Some users' preferred queries may have low scores in Candidate Generation but will be correctly ranked in Ranking. Therefore, users are more likely to find their preferred queries with a larger candidate set and thus result in a higher *CTR*. *response time* increases because more query candidates lead to more predictions in the Ranking stage, which is obviously more time consuming. Hence, we should carefully set the number of query candidates to trade off the efficiency and effectiveness. This also verifies the necessary of the Candidate Generation stage, since it is impossible to rank millions of queries in the Ranking stage within an acceptable time.

Our interactive RS has already gone into production on Taobao since Nov. 11, 2018, with about 4.5×10^7 active users per day. Users will see the interactive user interfaces after clicking some items and returning back to the homepage of Taobao App. The ranking model is daily updated, which is initialized by the parameters in the previous day and fine-tuned with the data obtained in the new day. In this way, the model can not only remember old data but also continuously fit the latest data to achieve better results.

6 CONCLUSION

In this paper, we propose a query-based interactive RS, which can accurately generate personalized questions and recommend items closely-related to users' feedback. To ensure high efficiency and remarkable effectiveness, we propose a model based on Meta-Path for the Candidate Generation stage and an adapted Attention-GRU model for the Ranking stage. Offline and online experiments verify the effectiveness of our interactive RS. In future work, we will explore more about the *Item Recommendation* module. Furthermore, we would try to generate questions by other objects (e.g. videos), besides queries, and recommend items according to users' behaviors on these objects to improve the user experience in interactive RS. In addition, hypergraph [33] for modeling heterogeneous relations in Candidate Generation, combinatorial optimization [9] for Ranking (i.e. generating 4 queries as a whole in Figure 2 instead of simply ranking them by scores), active learning methods [26, 27, 35] for the recommendation of cold-start queries are also interesting studies.

ACKNOWLEDGMENTS

Thanks to the colleagues in Alibaba Group. This research was supported by the National Natural Science Foundation of China (Grant Nos. 61936006, 61672409, 61876144, 61876145), the Major Basic Research Project of Shaanxi Province (Grant No. 2017ZDJC-31), Shaanxi Province Science Fund for Distinguished Young Scholars (Grant No. 2018JC-016), and the Fundamental Research Funds for the Central Universities (Grant Nos. JB190301, JB190305).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473* (2014).

[2] 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 *KDD*. ACM, 1187–1196.

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

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* (2014).

[5] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *NIPS*. 577–585.

[6] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. 2018. Q&r: A two-stage approach toward interactive recommendation. In *KDD*. ACM, 139–148.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Recsys*. ACM, 191–198.

[8] Mehdi Elahi, Matthias Braunhofer, Francesco Ricci, and Marko Tkalcic. 2013. Personality-based active learning for collaborative filtering recommender systems. In *Congress of the Italian Association for Artificial Intelligence*. Springer, 360–371.

[9] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q Zhu. 2019. Exact-K Recommendation via Maximal Clique Optimization. In *KDD*. ACM, 617–626.

[10] Chen He, Denis Parra, and Katrien Verbert. 2016. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications* 56 (2016), 9–27.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*. ACM, 241–248.

[13] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* 29 (2012).

[14] Zhipeng Huang, Bogdan Cautis, Reynold Cheng, and Yudian Zheng. 2016. Kb-enabled query recommendation for long-tail queries. In *IJCAI*. ACM, 2107–2112.

[15] Michael Jugovac and Dietmar Jannach. 2017. Interacting with recommenders: overview and research directions. *TiiS* 7, 3 (2017), 10.

[16] Myunghwan Kim and Jure Leskovec. 2013. Nonparametric multi-group membership model for dynamic networks. In *NIPS*. 1385–1393.

[17] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*. ACM, 661–670.

[18] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *KDD*. ACM, 1222–1230.

[19] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *NIPS*. 2204–2212.

[20] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[21] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. 2012. Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization. In *ICML*. 449–456.

[22] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. 2015. Active learning in recommender systems. In *Recommender systems handbook*. Springer, 809–846.

[23] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* 1 (2001), 285–295.

[24] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB* 4, 11 (2011), 992–1003.

[25] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: a review. *Social Network Analysis and Mining* 3, 4 (2013), 1113–1133.

[26] Beidou Wang, Martin Ester, Jiajun Bu, Yu Zhu, Ziyu Guan, and Deng Cai. 2016. Which to view: Personalized prioritization for broadcast emails. In *WWW*. International World Wide Web Conferences Steering Committee, 1181–1190.

[27] Beidou Wang, Martin Ester, Yikang Liao, Jiajun Bu, Yu Zhu, Ziyu Guan, and Deng Cai. 2016. The million domain challenge: Broadcast email prioritization by cross-domain recommendation. In *KDD*. ACM, 1895–1904.

[28] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA* 27 (2013).

[29] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*. ACM, 283–292.

[30] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Metagraph based recommendation fusion over heterogeneous information networks. In *KDD*. ACM, 635–644.

[31] Zhou Zhao, Ruihua Song, Xing Xie, Xiaofei He, and Yueting Zhuang. 2015. Mobile query recommendation via tensor function learning. In *IJCAI*. 4084–4090.

[32] Zhi-Dan Zhao and Ming-Sheng Shang. 2010. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE, 478–481.

[33] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.

[34] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*. 3602–3608.

[35] Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2019. Addressing the item cold-start problem by attribute-driven active learning. *TKDE* (2019).

[36] Yu Zhu, Junxiong Zhu, Jie Hou, Yongliang Li, Beidou Wang, Ziyu Guan, and Deng Cai. 2018. A Brand-level Ranking System with the Customized Attention-GRU Model. In *IJCAI*. ACM, 3947–3953.