

# Learning to Ask Critical Questions for Assisting Product Search

Zixuan Li\*  
e0348832@u.nus.edu  
Shopee Singapore Private Limited  
Sea-NExT Joint Lab  
National University of Singapore

Lizi Liao  
lzliao@smu.edu.sg  
Singapore Management University

Tat-Seng Chua  
dcscts@nus.edu.sg  
Sea-NExT Joint Lab  
National University of Singapore

## ABSTRACT

Product search plays an essential role in eCommerce. It was treated as a special type of information retrieval problem. Most existing works make use of historical data to improve the search performance, which do not take the opportunity to ask for user’s current interest directly. Some session-aware methods take the user’s clicks within the session as implicit feedback, but it is still just a guess on user’s preference. To address this problem, recent conversational or question-based search models interact with users directly for understanding the user’s interest explicitly. However, most users do not have a clear picture on what to buy at the initial stage. Asking critical attributes that the user is looking for after they explored for a while should be a more efficient way to help them searching for the target items. In this paper, we propose a dual-learning model that hybrids the best from both implicit session feedback and proactively clarifying with users on the most critical questions. We first establish a novel *utility* score to measure whether a clicked item provides useful information for finding the target. Then we develop the dual Selection Net and Ranking Net for choosing the critical questions and ranking the items. It innovatively links traditional click-stream data and text-based questions together. To verify our proposal, we did extensive experiments on a public dataset, and our model largely outperformed other state-of-the-art methods.

## CCS CONCEPTS

• **Applied computing** → **Electronic commerce**; • **Information systems** → **Users and interactive retrieval**; **Retrieval models and ranking**.

## KEYWORDS

product search, learning to ask, session-aware recommendation, interactive search

### ACM Reference Format:

Zixuan Li, Lizi Liao, and Tat-Seng Chua. 2022. Learning to Ask Critical Questions for Assisting Product Search. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom’22)*. ACM, New York, NY, USA, 9 pages.

\*Zixuan Li is under the Industrial Postgraduate Program supported by Singapore Economic Development Board, Shopee Singapore Private Limited and National University of Singapore.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).  
SIGIR eCom’22, July 15, 2022, Madrid, Spain  
© 2022 Copyright held by the owner/author(s).

## 1 INTRODUCTION

Online shopping provides a convenient and cost-effective shopping method for people over the Internet, which has ignited an increasing interest in eCommerce technologies. Product search is one of the main traffic sources that drives conversion. Users usually formulate queries to express their needs and find products of interest by exploring the retrieved results. The majority of such methods apply some similarity functions to match the query and documents that describes a product. Under such cases, the query and product documents are usually represented as vectors in observed or latent vector space to compute the distances [1, 9]. However, the search queries are often too general to capture the minute details of the exact product that the user is looking for [37].

Hence, there are two broad branches of research trying to address this problem. On one hand, session-aware recommendation makes use of the user’s behavior data within the session to capture user’s current interest, *i.e.*, predicts the successive item(s) that an anonymous user is likely to interact with. Two major classes of approaches are applied: Markov-based models and DNN-based models [19]. The former emphasizes on capturing sequential patterns such as in the popular FPMC method [23]. Such models often emphasize the last click in history while ignoring previous behaviors. Hence, they often adopt static representations of user intentions, which may result in sub-optimal performance. The later one employs deep learning models such as RNNs to capture users’ general preferences and current interests together. They leverage recurrent deep neural networks to encode historical interactions into hidden vector states, which leads to general and more informative representations. Nonetheless, there is still room for improvement since such models over-emphasize on monotonic behavior chains while failing to consider the more complex transition patterns among items.

On the other hand, interactive methods allow users to directly specify their needs in details. It is often realized under a conversational search and recommendation framework, where user preferences are clarified via aspect-value pairs [2, 35] or even unstructured text [33]. There are also efforts prompting users with clarification questions under the traditional search setting with a query [34, 37]. These methods largely enhance the accuracy of user preference modeling and flexibility of interaction. However, these conversational product search methods require intensive user involvement. They assume that the user has a clear target in mind, which does not hold true especially in early browsing stages [10].

In this work, we propose **DualSI** – a dual-learning model to hybrid the best from both session-aware recommendation and interactive methods. It makes use of the implicit session feedback to model user interests and proactively ask users the most critical questions to shorten the exploring session. Different from the traditional *relevance* score widely used in product search, we first

establish a new *utility* score to measure how much useful information a clicked item provides for finding the target in a given context. Then we develop a Transformer-based selection module to calculate such contextualized *utility* score which is able to capture the complex transition patterns beyond monotonic chain patterns. By further linking the most critical questions to the highest *utility* scores, a dual ranking model predicts the target item based on the user responses. We conduct extensive experiments on a public dataset, and it shows that the proposed DualSI model largely outperforms baselines from both session-aware recommendation and interactive search.

To sum up, the main contributions of this work are as follows:

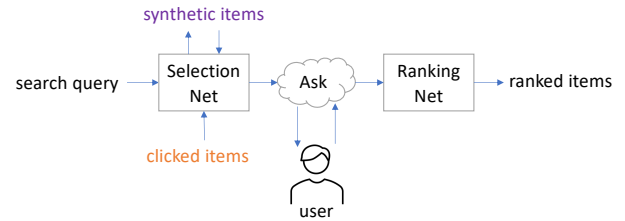
- (1) We explicitly define a novel *utility* score which measures whether a clicked item provides useful information in certain context for finding the target product.
- (2) We design a dual-learning framework that incorporates a Transformer-based selection module for contextualized *utility* score calculation and a dual ranking module for target item prediction.
- (3) Extensive experiments demonstrate the effectiveness of our proposed method. Qualitative analysis results also show that the proposed method not only asks the right question for target finding but also helps to shorten the exploring session for users.

## 2 RELATED WORK

In this section, we provide a brief overview of the existing research efforts closely related to our work. Making use of user's behavior data within the session as implicit feedback for understanding user's current preferences is commonly known in session-aware recommendation. The idea has also been used in a few search related works. We will discuss them together under the session-aware recommendation subsection. They are highly relevant as our work also allows users to explore first and uses the within-session implicit feedback as input. Another important topic is interactive search. Our proposed method also learns to efficiently interact with users via question answering to improve search quality. Furthermore, we also provide an overview about works regarding learning to ask, and briefly discuss about their connections to our work.

### 2.1 Session-aware Recommendation

Session-aware recommendation predicts the successive item(s) that an anonymous user is likely to interact with, according to the implicit feedback within the session [6, 22, 30]. Generally speaking, there are two major classes of approaches to leverage sequential information from users' historical records: Markov-based models and DNN-based models. In early days, Markov chains are widely applied to capture sequential patterns between consecutive user-item interactions [26]. For example, it was applied to characterize users' latest preferences with the last click, but the previous clicks and the useful information in the long sequence are neglected [25]. Rendel *et al.* proposed a hybrid model FPMC [23], which combined Matrix Factorization and Markov Chain to model sequential behaviors for next basket recommendation. However, the adoption of static representations for user intentions is a major problem of such methods.

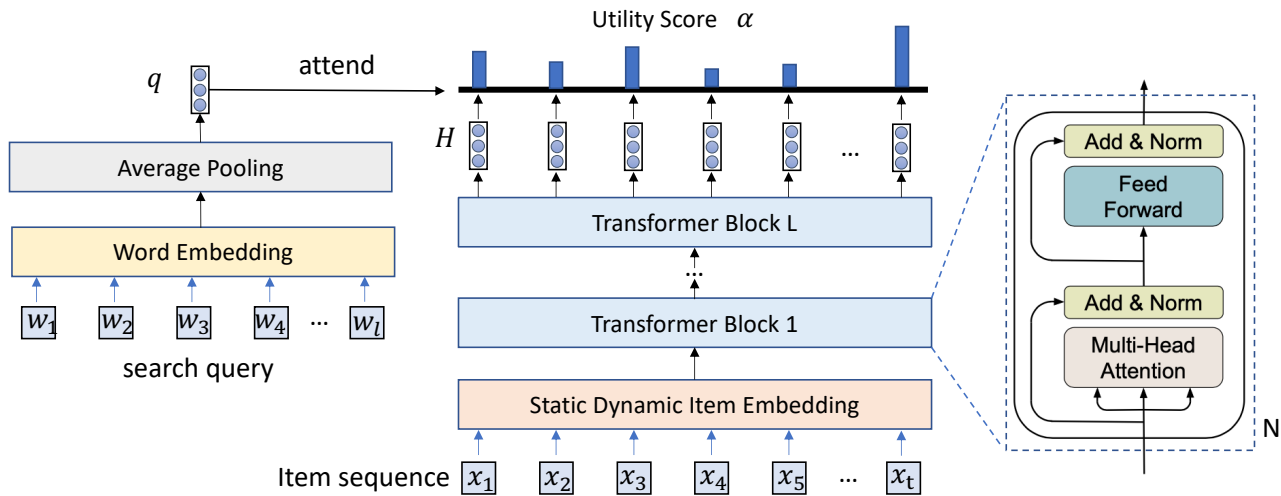


**Figure 1: The DualSI framework. The Selection Net calculates contextualized utility scores for clicked items, which helps to select the most critical questions. The dual Ranking Net learns to predict the target item based on user's feedback.**

More recently, the DNN-based models such as Recurrent Neural Networks (RNNs) have been employed in session-aware recommendation and demonstrated state-of-the-art performance. They are widely used to capture users' general interests and current interests together through encoding historical interactions into a hidden state (vector). The method in [12] is among the pioneer works that proposed a deep RNN-based model to encode items sequences for recommendation. Jing *et al.* [16] further improved it through adding extra mechanism to tackle the short memory problem inherent in RNNs. In addition, to explicitly model the effects of users' current actions on their next moves, the model in [18] utilized an attention net to model user's general states and current states separately. Besides, Hidasi *et al.* proposed two variations to improve their model performance through adjust loss functions [11, 13]. There are also works such as [31] adopting Graph Neural Networks (GNNs) to capture the complex transition patterns among the items in a session rather than the transition pattern of single way. Beside recommendation, some search work also incorporate the session-aware concept. Bi *et al.* makes use of implicit feedback for re-ranking the search results in the following page [3]. However, all these methods rely on inferred user interests or preferences which might be wrong or inaccurate. In this paper, we allow the model to solicit user feedback explicitly via asking critical questions.

### 2.2 Interactive Search

Interactive search puts human in the loop whereby user's preferences of a specific query can be directly captured [7, 15, 28]. Zhang *et al.* proposed a conversational search and recommendation framework and implemented it using a Multi-Memory Network architecture [35]. In this work, the product search is done after clarifying user's preferences on different aspect-value pairs via dialogue. Considering the imperfection of product attribute schema which has been commonly used for questions generation in conversational search [2, 35], Xiao *et al.* introduced an end-to-end conversational product search system that also incorporates unstructured text to enhance the product knowledge [33]. Other than purely focusing on the textual modality, there are also works incorporating images into the dialogue [17, 20]. However, all these conversational product search approaches require intensive user involvement and assume the users already have specific item in mind which might not be true. Shoppers usually do not have a clear picture of the item in mind when they are at early shopping phase [4, 10].



**Figure 2: The Selection Net architecture. The item sequence obtains contextualized representations via static dynamic item embeddings and self-attention blocks. The query signals attend over them for better utility score calculation. Details of the Static Dynamic Item Embedding module are further provided in Figure 3.**

Another branch of work for interactive search prompts users with clarification questions in the traditional search setting [34, 37]. After user issuing the query, the clarification question is asked with options provided for the user to interact with. Zou *et al.* introduced a Bayesian-based product search method which sequentially prompts users on their expected entities in the target product document [37]. A fixed number of questions are asked one by one whereby the next question is generated based on the user’s answers on previous questions. However, their proposed algorithm can only handle seen queries. Similarly, [36] makes use of historical user ratings for offline initialization and learns to ask a sequence of questions so as to understand user preferences for recommending items. Other than text query, in [27], it asks user to click on his/her interested product query image region, and use the information for generating more accurate results that targets the user’s specific interest. These work also have similar problems as mentioned for conversational search above. Our work avoids these issues by only asking the most critical questions on top of the observed user click behaviors.

### 2.3 Learning to Ask

Learning to ask aims to seek enough information for specific tasks with as few questions as possible. Generally speaking, it is naturally involved in all conversational and user interaction related works. Some papers studied this topic specifically in a 20 questions game setting, whereby one party think of an object and the other party tries to guess by asking no more than 20 questions. For example, Wu *et al.* introduced an entropy-based ranking algorithm to calculate the object-question relevance matrix utilizing user’s answer distributions [32]. Almost in the same time period, in [5], the authors used deep reinforcement learning and general matrix factorization for the agent to learn smart questioning strategies. Similarly, Hu *et al.* proposed a policy-based reinforcement learning method that enables the agent to learn optimal policies via continuous interaction

with user [14]. However, these works require heavy simulation for the agent to learn well, which has been avoided in our work by leveraging user’s implicit feedback.

## 3 METHOD

This section provides a detailed description of the proposed method. The overall framework is shown in Figure 1. Our approach consists of two parts: (a) the calculation of utility score using Selection Net for critical question selection; and (b) the product ranking using the dual Ranking Net given user’s within session behavior and answers to the critical questions. This design leverages both user’s implicit feedback and explicit question answering, which is inspired by the fact that shoppers usually gets clearer on the exact item to purchase after some initial browsing, and directly clarifying with the user afterwards on the selected critical questions would efficiently improve the shopping experience. The detailed training scheme is also presented in the end of this section.

### 3.1 Contextualized Selection Net

The core of our method is the contextualized Selection Net. It takes the click-stream data as input for context capturing and question selection via the utility score.

#### 3.1.1 Utility Score.

Search queries are usually very general, especially when a user does not have a clear picture on what to buy or cannot describe precisely the item he/she is looking for. After issuing the search query, a user usually needs to browse for a while on different products to eventually find the ‘right’ item. During the browsing process, the user’s idea on what exactly to buy becomes clearer. Each clicked item in the click sequence carries different level of information on helping the user to find the target item. We explicitly measure it by introducing the utility score  $\alpha$ . Intuitively, higher utility score

implies that the item carries more critical information related to the target product within its search context.

In our implementation, the utility score is instantiated by calculating the attention score between the search query  $q$  and contextualized hidden states of the clicked items  $H = [h_1, h_2, \dots, h_t]$  as indicated in Figure 2. We design the loss function to enable the training of the Selection Net to learn to generate higher scores for clicked items that carries more information leading to the target. We hypothesize that such ‘information’ is expressed by the item attributes, whereby each product can be uniquely described using a set of attributes. Such ‘information’ is also reflected in the relative occurrence times of attributes inside the click stream. This is intuitive as these repeatedly clicked items/attributes are usually more interested by the user. In the inference stage, the synthetic items mentioned in Figure 1 are generated using different combinations of attributes. By inputting each of them together with the clicked items through the Selection Net, the synthetic item with highest utility score is chosen. Its attributes are used as critical questions to clarify with the user, and the hidden states  $H'$  after the last attention layer are then used in the ranking stage. More design and implementation details are covered in the next subsection.

### 3.1.2 Contextualized Selection.

With the concept of utility score being introduced, we now explain how the Selection Net is designed. As shown in Figure 2, to capture the complex search and browsing context, we encode the clicked items with static dynamic item embedding layer first, and then pass through a transformer encoder which can access information of all clicked items in the sequence to obtain the contextualized representations. The search query  $q$  then attend on the output of the transformer module (i.e. hidden states  $H$ ). More attention should be given to the items with higher utility score. Hence, we use the attention score directly. The ground truth score for the target item is 1, as the target item is ideally the one carries the highest utility score. The scores of other items are suppressed.

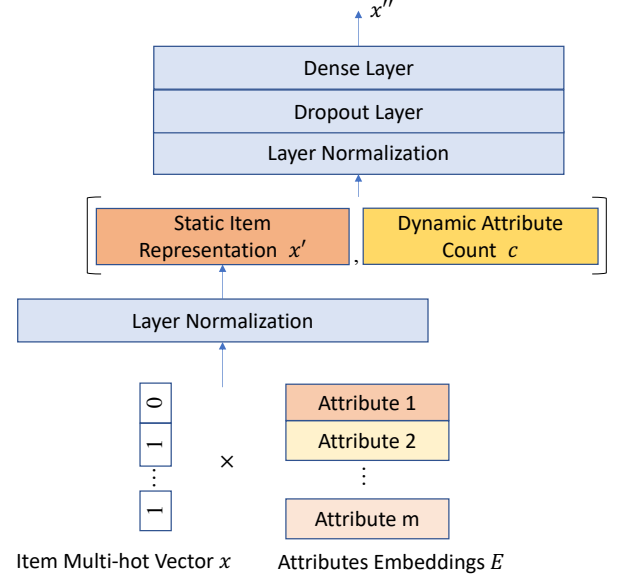
**Query Embedding.** As suggested in [3], more complex encoding approaches are not expected to have a better performance than the simple average. A query issued by an user typically consists of multiple words, where the word embedding  $\varepsilon(w) \in \mathbb{R}^d$  and  $d$  is the embedding dimension. In our implementation, we use the widely applied GloVe word embeddings [21]. We average the word embeddings to obtain the corresponding query representation, i.e.,

$$q = \frac{\sum_{w \in \text{query}} \varepsilon(w)}{l} \quad (1)$$

where  $l$  is the number of words in the query.

**Static Dynamic Item Embedding.** Each item can be uniquely represented by the set of attributes associated with it. We firstly create a multi-hot vector  $x \in \mathbb{R}^{m \times 1}$  for each item, where  $m$  refers to the number of distinct attributes of the product pool. With the corresponding attribute positions flagged as 1 in  $x$ , the static dense representation for an item is

$$x' = LN(x^T E) \quad (2)$$



**Figure 3: The static item embedding concatenate with the dynamic attribute count feature to form a comprehensive representation for each item.**

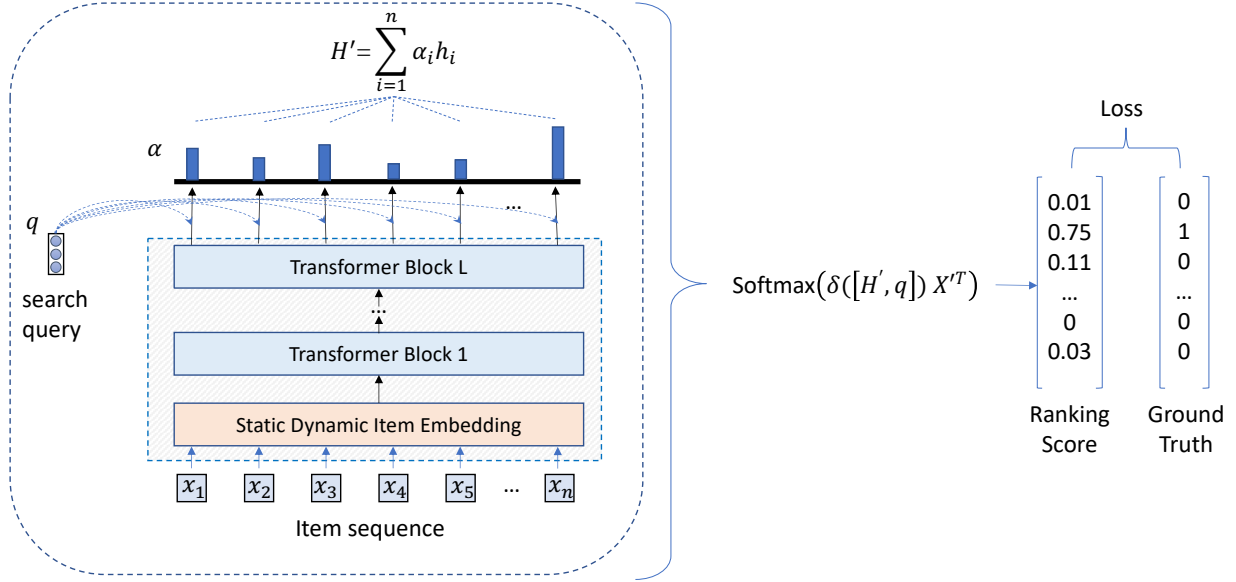
where  $E = \begin{bmatrix} e_1 \\ \dots \\ e_m \end{bmatrix}$  and  $e_m$  is the embedding for the  $m$ -th attribute generated in a similar way as query embedding.  $LN(\cdot)$  refers to the layer normalization operation.

On top of the static item embedding mentioned above, we concatenate it with a dynamic feature called Attribute Count  $c$  before inputting it to the transformer encoders:

$$x'' = Dense(Dropout(LN([x', c]))) \quad (3)$$

where  $Dropout(\cdot)$  refers to the dropout layer and  $Dense(\cdot)$  refers to the fully-connected feed-forward layer. It is intuitive to assume that attributes appeared more frequently in the clicked item sequence are more relevant to the target item. This is also verified by looking at its capability in finding target items in our preliminary analysis, hence it is used here explicitly. The Attribute Count feature is a simple average of the accumulated appearance counts for each flagged attribute associated with the item by the time it is clicked. After the concatenation, a dynamic representation is obtained with dimension  $d + 1$ . To normalize across the features, layer normalization is used. In addition, a dropout layer and a dense layer is applied which projects it to dimension  $d$ . An illustration on how the dynamic representation is obtained for each input item can be found in Figure 3. We will also show how the popular attributes affect the learning-to-ask performance in the ablation study later.

**Contextualized Representation.** Now, we map the comprehensive representation of the sequence of clicked items to contextualized embeddings. Specifically, we map these into  $H = [h_1, h_2, \dots, h_t]$  using  $L$  transformer encoder layers following the definition in [29]. We adopt the transformer module because of its capability in capturing the complex context information among the clicked items.



**Figure 4: The Dual Ranking Net.** There are normalization layer, flatten layer, dropout layer and dense layer used in the  $\delta$  network. After multiplying with the item pool  $X'^T$ , there are again a normalization layer, some dense layers and dropout layers. They are omitted here for a concise and clear illustration on the main concept.

In the actual implementation, we set  $L=1$  as the experiment dataset is not too large to model. It can be adjusted accordingly for other use cases. The queries  $Q$ , keys  $K$  and values  $V$  are all referring to the linear transformations of  $x''_{1:t}$  in the case of self-attention in transformer encoder

$$H = \text{TransformerEncoder}(x''_{1:t}) \quad (4)$$

Inside the transformer encoder, we use four heads for multi-head attention.

$$\text{MultiHead}(x''_{1:t}) = [\text{head}_1, \text{head}_2, \text{head}_3, \text{head}_4]W^O \quad (4.1)$$

$$\text{head}_i = \text{Attention}(x''_{1:t}W_i^Q, x''_{1:t}W_i^K, x''_{1:t}W_i^V) \quad (4.2)$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V \quad (4.3)$$

where  $W^O$ ,  $W_i^Q$ ,  $W_i^K$  and  $W_i^V$  are the projection parameter matrices. It is then followed by residual connections, layer normalizations and dense layers as in the original transformer paper [29], except we set  $N=1$  for relatively small data size.

**Utility Score.** The attention layer calculates the utility score for each item with contextualized  $H$  and the query  $q$ :

$$\alpha_i = \frac{\exp(q \cdot h_i^T)}{\sum_{j=1}^t \exp(q \cdot h_j^T)} \quad (5)$$

**Loss Function.** During training, the utility score  $\alpha$  is maximized at the target item position by minimizing the mean absolute error

(MAE) loss, i.e.,

$$\mathcal{L}_{\text{selection}} = \frac{\sum_{j=1}^t |\alpha_j - y_j|}{t} \quad (6)$$

where  $y_j$  is the ground truth label at position  $j$ .

The training of the Selection Net aims to gain the capability of identifying the item with the highest utility score, i.e. the target item. During testing, given that  $k$  attributes are to be asked at one question prompt, each  $k$ -attributes combination is encoded into the multi-hot vector to form a synthetic item  $x_{\text{synthetic}}$ , and then pass through the trained Selection Net together with the clicked items, i.e.  $x_{1:t-1}$ . The one with the highest utility score in the given click sequence is chosen to prompt to the user for feedback.

### 3.2 The Dual Ranking Net

With the given user behavior context and answers to the questions, the dual Ranking Net learns to rank the items. It reuses the parameters from the trained Selection Net as shown in Figure 4 (the shaded encoding part is frozen during training). After encoding the item sequence, we obtain the contextualized embedding  $H'$  from the encoder net

$$H' = \sum_{i=1}^n \alpha_i h_i \quad (7)$$

where  $n$  is the item sequence length including the synthetic item from selected questions.  $n$  equals to  $t$  only when the system prompts user questions one step before the target.

For the product relevance ranking,  $H'$  is firstly concatenated with the search query  $q$ . Then the combined representation goes through



a  $\delta$  network which consists of a normalization layer, a flatten layer, a dense layer and a dropout layer. Then it is multiplied with the item pool  $X'^T$ , where  $X' = [x'_1, x'_2, \dots, x'_p]$  and  $p$  is the number of products. On top of that, it further connects to a normalization layer and a two-layer feed-forward network. A softmax function is applied as the last layer to get the ranking score of each item.

$$\text{score}(X') = \text{Softmax}(\delta([H', q])X'^T) \quad (8)$$

The cross entropy loss  $\mathcal{L}_{\text{ranking}}$  is calculated between the predicted scores and ground truth  $y'$  for the Ranking Net training.

$$\mathcal{L}_{\text{ranking}} = \text{CrossEntropy}(\text{score}(X'), y') \quad (9)$$

### 3.3 Training and Inference Scheme

Below is a detailed explanation on how the training and inference are done with the Selection Net and Ranking Net.

#### 3.3.1 Training Stage.

For Selection Net, it takes the sequence of clicked items up to the target item for training as shown in Figure 2. When it comes to Ranking Net, the item sequence is formed by the clicked items  $x_{1:t-1}$  together with each of the synthetic items generated using the attributes appeared in the target item. After getting the utility score for each composed sequence, only the one with maximum utility score at the synthetic item position is passed through to the rest of the Ranking Net for training. An illustration can be found in Figure 5. Padding and Masking techniques are used to standardize the input for different click sequence length.

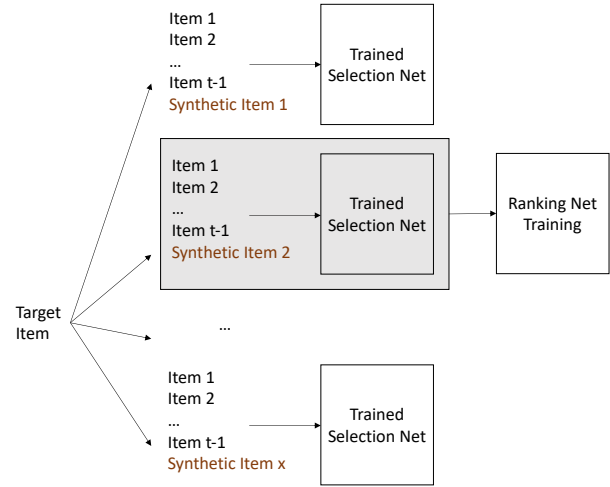
#### 3.3.2 Inference Stage.

During testing stage, instead of generating synthetic items from target item attributes, they are generated from all possible combinations of the attribute pool, as it is for the Selection Net to decide on which question to ask. Each of these synthetic item combines with the  $n$  clicked items, and input into the trained Selection Net. The one with highest utility score is then used for generating the critical questions. If the attribute asked exist in the target item, it simulates that the user answered with 'Yes' to this attribute. Based on the answer, the Ranking Net then takes the corresponding  $H'$  for relevance ranking. For cases where the user answered with 'No' or answered 'Yes' to some of the attributes only, we gets the input sequence updated accordingly for getting the  $H'$  for eventual ranking.

## 4 EXPERIMENTS

In this section, we conduct extensive experiments mainly to answer the following research questions:

- RQ1** Our proposed method leverages both implicit feedback and explicit feedback. Is it performing better than those using only one of them?
- RQ2** Does our method ask the right questions? Are the answers from users play an important role?
- RQ3** Would the proposed method manage to shorten the search sessions for user?



**Figure 5: Training scheme for the Ranking Net. The one in the grey box is chosen as the input for Ranking Net Training if synthetic item 2 has the highest score among all synthetic items generated based on the target item.**

## 4.1 Experimental Setup

### 4.1.1 Dataset.

We use the Diginetica dataset<sup>1</sup> which was first introduced in 2016 for CIKM Cup on product search. It consists of query-full and query-less search related data. Query-full search refers to those triggered by a user issued query, while query-less search refers to those triggered by a category click. We only used the query-less search data as the amount of query-full search data is very small after processing. For query-less search data we are using, the category ID is used to represent the search query, which is a common practice in many product search works [24, 37]. We also filter out search sessions with less than 5 or more than 20 clicks before purchase. In principle, our DualSI can handle any sequence length. This preprocessing is mainly for a standard evaluation comparison later. After random split, 75% of the data is used for training, 10% for validation and 15% as test set. To the best of our knowledge, this is the only public dataset that matches our setting.

### 4.1.2 Question Pool Construction.

The question pool can be constructed by extracting attributes from product documents, which includes but not limit to product title, product description, product price range and etc. There are different methods for extracting product attributes from its document, such as using the TAGME tool [8]. In the Diginetica dataset we use, since only hashed product title and price are available, we took all the available words in these as possible attributes. Due to data limitation, the minimum number of attributes for each item is two. This is why we only did experiment on asking up to two questions. Nevertheless, this aligns with our aim of only asking critical questions without bothering the user too much. In addition, we perform simple pre-processing such as filtering out attributes that only appeared once.

<sup>1</sup><http://cikm2016.cs.iupui.edu/cikm-cup>

**Table 1: Performance comparison regarding metrics such as MRR, Top-3 Accuracy and NDCG among baselines and variants of our model.**

	No Question Asking			Asking 1 Question/Attribute			Asking 2 Questions/Attributes		
	MRR ↑	Top-3 Accuracy ↑	NDCG ↑	MRR ↑	Top-3 Accuracy ↑	NDCG ↑	MRR ↑	Top-3 Accuracy ↑	NDCG ↑
SR-GNN	0.08606	0.09105	0.19237	-	-	-	-	-	-
SCEM	0.16187	0.18681	0.27548	-	-	-	-	-	-
QSBPS	-	-	-	0.15971	0.16169	0.25736	0.18476	0.20251	0.28644
Qrec	-	-	-	0.16746	0.16908	0.25596	0.17567	0.17713	0.26595
DualSI_randomQ	-	-	-	0.16015	0.17268	0.27312	0.16015	0.17268	0.27314
DualSI_randomA	-	-	-	0.19315	0.21821	0.30364	0.17613	0.19623	0.28722
DualSI_popular	-	-	-	0.22443	0.25275	0.33162	0.24273	0.27630	0.34705
DualSI	0.16015	0.17268	0.27313	<b>0.23310</b>	<b>0.25746</b>	<b>0.33909</b>	<b>0.24751</b>	<b>0.28571</b>	<b>0.35075</b>

**Table 2: Performance comparison of the proposed DualSI method among the settings of asking question 1-5 item clicks before the actual target item is clicked.**

	Asking 1 Question/Attribute			Asking 2 Questions/Attributes		
	MRR ↑	Top-3 Accuracy ↑	NDCG ↑	MRR ↑	Top-3 Accuracy ↑	NDCG ↑
1 click	0.23310	0.25746	0.33909	0.24751	0.28571	0.35075
2 clicks	0.21790	0.24333	0.32586	0.23178	0.25432	0.33736
3 clicks	0.19721	0.22449	0.30584	0.19778	0.21350	0.30801
4 clicks	0.15445	0.16797	0.26799	0.16688	0.18838	0.28080
5 clicks	0.13399	0.14443	0.24608	0.14575	0.16484	0.25755

#### 4.1.3 Evaluation Measures.

To evaluate our method, we use Mean Reciprocal Rank (MRR), Top-3 Accuracy and Normalized Discounted Cumulative Gain (NDCG) as the evaluation metrics. In our setting, there is only one target product for each purchase, which makes MRR a proper metric to use to calculate the inversed rank value of the target item. The Top-3 Accuracy suggests whether the target item appears in the top-3 ranked items. NDCG is another important metric that is commonly used for product ranking performance evaluation, as it can capture the graded relevance values. Although the relevance is binary in our task setting, it is still a good indicator on how well the products are ranked relative to the ideal ranking.

#### 4.1.4 Baselines.

To verify the effectiveness of our model DualSI and answer the three research questions, we compare it with several baselines.

- **QSBPS** [37]. The QSBPS model uses a Bayesian approach for sequentially asking users on their expected attributes. It represents the interactive search methods that queries user for explicit feedback. Specifically, the model directly queries the users on the expected presence of entities in the relevant product documents. It is based on the assumption that there is a set of candidate questions for each product to be asked.

- **Qrec** [36]. Qrec is an interactive recommendation method which recommend items to users by asking their preferences in a similar way as QSBPS. It is initialized offline with a matrix factorization model using the historical item ratings by users, followed by an online update of user and item latent factors based on user’s answer. It learns to select the best question sequence to ask. Since there is no rating data in our dataset, we treat each purchase pair with a rating equal to 3.
- **SR-GNN** [31]. SR-GNN is one of the state-of-the-art model for session-aware recommendation. It constructs the clicked items into graph to capture the complex transitions, and predicts the next item to be clicked. Although it is not specifically designed for search task, it can be used to represent those session-aware methods which leverage implicit feedback from user clicks. In our setting, the next item refers to the target item.
- **SCEM** [3]. The SCEM model makes use of the clicked item sequence for search result re-rank. Specifically, it leverages clicks within a query session, as implicit feedback, to represent users’ hidden intents, which further act as the basis for re-ranking subsequent result pages for the query. We use it together with SR-GNN as baselines representing the product search methods that solely use click-stream data.

- **DualSI** variations. The first variant DualSI\_randomQ randomly selects attributes to ask on top of our general framework. Similarly, DualSI\_randomA randomly answers the questions asked. In DualSI\_popular, questions are generated based on the most frequently appeared attributes in the clicked items.

## 4.2 Main Results (RQ1)

As shown in Table 1, our method largely outperforms the baselines across all metrics. When asking one or two questions are allowed, the proposed DualSI method outperforms QSBPS and Qrec by large margins. For example, it obtains 45.96% and 33.96% performance gain comparing with QSBPS regarding MRR respectively. It also outperforms the SR-GNN and SCEM which only considers user click-stream data. We show that making use of the implicit session feedback to model user interests while allowing question asking is indeed an effective way to assist the users. This addresses the **RQ1**.

Besides, asking two questions/attributes helps to gather more information from the users, hence the performance is better. This is as expected for both the QSBPS and Qrec baselines, and the proposed model and its variations.

The DualSI\_popular achieves relatively good performance as compared to other variations. It uses the most frequently appeared attributes among the clicked items for question generation. This result suggests that the key attributes are usually unchanged though the click path and they play an essential role on critical question asking. The interpretation of the other variations will be discussed in the next subsection.

We also observe that when no question asking is allowed, our method performs slightly worse than SCEM. This might be due to the fact that the Ranking Net of DualSI shares encoder with the Selection Net while the later one is trained to select critical questions. When no question is asked, the encoder part deviates from its original learning purpose and hence affect the ranking results. As for SR-GNN, it performs the worst. This might be because SR-GNN is not specifically designed for search task and does not take query into consideration. Moreover, it does not take any textual information of the items.

## 4.3 Qualitative Analysis

### 4.3.1 The Effect of Questions Asked (RQ2).

Two model variants, DualSI\_randomQ and DualSI\_randomA, are explored to answer **RQ2**. We use the same training settings for each method, keeping all the components the same except the question part. As it is shown in Table 1, the experiment results on randomly generating questions (DualSI\_randomQ) are similar to that of SCEM and DualSI (No Question Asking) which both make use of item click data only. This is generally to be expected. Besides, since the questions are randomly generated, the number of attributes being asked does not matter too much and there is limited performance difference on asking one question or two questions. The large performance gap between this variant and our original model suggests the effectiveness of the questions selected. It validates our design of the utility score and the effectiveness of the transformer-based Selection Net structure.

The other variant DualSI\_randomA randomly generates answers to the questions asked. As our Selection Net guarantees the quality of the questions asked, randomly answering 'Yes' or 'No' still maintains the performance at an acceptable level. This might be due to the fact that randomly answer the questions may still obtain an approximately 50% chance to guess it right. Therefore, the result obtained is better than that of DualSI\_randomQ. It is interesting to note that, under the variant DualSI\_randomA, the performance on asking 2 attributes performs worse than asking 1 attribute. After carefully inspecting the experimental instances, we suspect that it is probably because of the increased noise introduced when randomly answering the acceptance/rejection on more attributes. From a different angle, this actually suggests the importance of gathering the correct answers from the users to directly know their preferences. This would increase the chance of getting the correct target item with less operations and save users' time.

### 4.3.2 The Position for Prompting Questions (RQ3).

We also conducted experiments on promoting question at different positions of the user's browsing path. The experimental results are presented in Table 2. On one hand, by asking at a relatively later position in the browsing path, the performance of the model increases accordingly. This aligns with our expectation, as seeing more clicked items would provide a more representative context and assist on asking more critical questions accurately. On the other hand, despite of asking the questions early (1-3 item clicks before target item), our model still outperforms the baselines. This suggests that our method can help to shorten the user search sessions. In real application, it is possible to find a good balance between ranking performance and session shortening.

## 5 CONCLUSION

In this work, we proposed a dual-learning model **DualSI** which consists of a Selection Net and a dual Ranking Net. It leverages both implicit user feedback from user's click-stream data and explicit user feedback by asking critical questions. Note that existing product search methods either cannot capture user's current detailed interest or require intensive user involvement for clarifications while most users do not have an exact item in mind at the initial stage. It is more appropriate to allow users to browse first and then clarify with him/her on the target item. Hence, we introduced the innovative idea of utility score in Selection Net which helps to select the most critical questions to ask by learning from the clicked items. After incorporating user's answer on the question, the dual Ranking Net then ranks the items. According to the experiment results on the public dataset, our model largely outperformed the state-of-the-art baselines.

In the future work, we plan to make use of the rejected entities also for performance improvement. Moreover, instead of promoting questions at a fixed position, dynamically prompting questions when needed could be an interesting direction to continue exploring.

## ACKNOWLEDGEMENT

This research is supported by Shopee Singapore Private Limited, Singapore Economic Development Board and Sea-NExT Joint Lab.



## REFERENCES

- [1] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 645–654.
- [2] Keping Bi, Qingyao Ai, Yongfeng Zhang, and W. Bruce Croft. 2019. Conversational Product Search Based on Negative Feedback. *Proceedings of the 28th acm international conference on information and knowledge management* (Nov 2019).
- [3] Keping Bi, Choon Hui Teo, Yesh Dattatreya, Vijai Mohan, and W. Bruce Croft. 2019. Leverage Implicit Feedback for Context-aware Product Search. In *eCOM@SIGIR*.
- [4] Chun-An Chen. 2009. Information-oriented online shopping behavior in electronic commerce environment. *J. Softw.* 4, 4 (2009), 307–314.
- [5] Yihong Chen, Bei Chen, Xuguang Duan, Jian-Guang Lou, Yue Wang, Wenwu Zhu, and Yong Cao. 2018. Learning-to-Ask. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery.
- [6] Minjin Choi, Jinhong Kim, Joonseok Lee, Hyunjung Shim, and Jongwuk Lee. 2021. Session-Aware Linear Item-Item Models for Session-Based Recommendation (*WWW '21*). Association for Computing Machinery, New York, NY, USA, 2186–2197.
- [7] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) (*KDD '16*). Association for Computing Machinery, New York, NY, USA, 815–824.
- [8] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-Fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*. 1625–1628.
- [9] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive long short-term preference modeling for personalized product search. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–27.
- [10] Tobias Hatt and Stefan Feuerriegel. 2020. Early detection of user exits from click-stream data: A markov modulated marked point process model. In *Proceedings of The Web Conference 2020*. 1671–1681.
- [11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.
- [12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2016).
- [13] Balázs Hidasi, Massimo Quadran, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 241–248.
- [14] Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. 2018. Playing 20 Question Game with Policy-Based Reinforcement Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- [15] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. *Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems*. Association for Computing Machinery, New York, NY, USA, 304–312.
- [16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.
- [17] Lizi Liao, Yunshan Ma, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2018. Knowledge-Aware Multimodal Dialogue Systems. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) (*MM '18*). Association for Computing Machinery, New York, NY, USA, 801–809.
- [18] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.
- [19] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1091–1100.
- [20] Seungwhan Moon, Satwik Kottur, Paul Crook, Ankita De, Shivani Poddar, Theodore Levin, David Whitney, Daniel Difrancia, Ahmad Beirami, Eunjoon Cho, Rajen Subba, and Alborz Geramifard. 2020. Situated and Interactive Multimodal Conversations. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 1103–1121.
- [21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
- [22] Tu Phuong, Tran Thanh, and Ngo Xuan Bach. 2019. Neural Session-Aware Recommendation. *IEEE Access* PP (07 2019), 1–1. <https://doi.org/10.1109/ACCESS.2019.2926074>
- [23] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 1–22.
- [24] Jennifer E. Rowley. 2000. Product search in e-shopping: a review and research propositions. *Journal of Consumer Marketing* 17 (2000), 20–35.
- [25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.
- [26] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).
- [27] Tomáš Škopal, Ladislav Peška, and Tomáš Grošup. 2018. Interactive product search based on global and local visual-semantic features. In *International Conference on Similarity Search and Applications*. 87–95.
- [28] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) (*SIGIR '18*). Association for Computing Machinery, New York, NY, USA, 235–244.
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in neural information processing systems*. 5998–6008.
- [30] Chen Wu, Ming Yan, and Luo Si. 2017. Session-aware Information Embedding for E-commerce Product Recommendation. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017).
- [31] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.
- [32] Momo Klyen Kyohei Tomita Xianchao Wu, Huang Hu and Zhan Chen. 2018. Q20: Rinna riddles your mind by asking 20 questions. In *Japan NLP*.
- [33] Liqiang Xiao, Jun Ma, Xin Luna Dong, Pascual Martínez-Gómez, Nasser Zalmout, Wei Chen, Tong Zhao, Hao He, and Yaohui Jin. 2021. End-to-End Conversational Search for Online Shopping with Utterance Transfer. *CoRR* abs/2109.05460 (2021). arXiv:2109.05460
- [34] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating Clarifying Questions for Information Retrieval. In *Proceedings of The Web Conference 2020*.
- [35] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM international conference on information and knowledge management*. ACM, 177–186.
- [36] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards Question-based Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery.
- [37] Jie Zou and Evangelos Kanoulas. 2019. Learning to ask: Question-based sequential Bayesian product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 369–378.