Generating Clarifying Questions with Web Search Results

Ziliang Zhao
Zhicheng Dou
Jiaxin Mao
zhaoziliang@ruc.edu.cn
dou@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Ji-Rong Wen
jrwen@ruc.edu.cn
Beijing Key Laboratory of Big Data Management and Analysis Methods
Beijing, China
Key Laboratory of Data Engineering and Knowledge Engineering, MOE
Beijing, China

ABSTRACT

Asking clarifying questions is an interactive way to effectively clarify user intent. When a user submits a query, the search engine will return a clarifying question with several clickable items of sub-intents for clarification. According to the existing definition, the key to asking high-quality questions is to generate good descriptions for submitted queries and provided items. However, existing methods mainly based on static knowledge bases are difficult to find descriptions for many queries because of the lack of entities within these queries and their corresponding items. For such a query, it is unable to generate an informative question. To alleviate this problem, we propose leveraging top search results of the query to help generate better descriptions because we deem that the top retrieved documents contain rich and relevant contexts of the query. Specifically, we first design a rule-based algorithm to extract description candidates from search results and rank them by various human-designed features. Then, we apply an learning-to-rank model and another generative model for generalization and further improve the quality of clarifying questions. Experimental results show that our proposed methods can generate more readable and informative questions compared with existing methods. The results prove that search results can be utilized to improve users' search experience for search clarification in conversational search systems.

CCS CONCEPTS

• **Information systems** → **Search interfaces**.

KEYWORDS

Clarifying Question, Search Clarification, Conversational Search

ACM Reference Format:
Ziliang Zhao, Zhicheng Dou, Jiaxin Mao, and Ji-Rong Wen. 2022. Generating Clarifying Questions with Web Search Results. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531981

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531981

1 INTRODUCTION

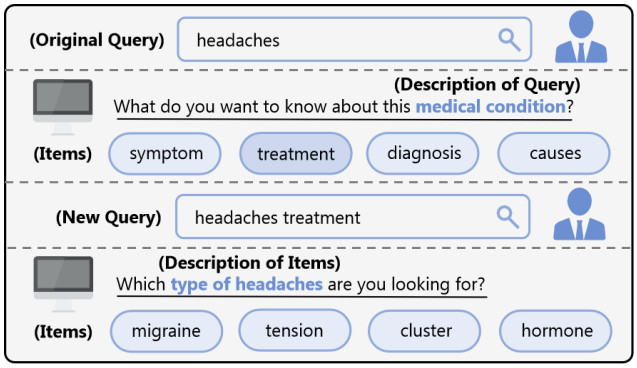

Figure 1: Search clarification in a conversational search system. In this paper, we focus on generating better clarifying questions (the text underlined), other than generating items.

User's queries can be ambiguous or faceted [6], making search engines difficult to understand the user's information needs. For example, the ambiguous query "*apple*" could indicate Apple company or a kind of fruit, and the faceted query "*lost*" has various aspects like seasons and characters. In this circumstance, users often need to scan results one by one or reformulate a new query to retrieve a new result list. Without any information guidance, this time-consuming process greatly impacts users' search experience.

Currently, several methods in Information Retrieval (IR) can help users articulate their information needs, like search result diversification [34], query suggestion and refinement [2], and query facets mining [10]. With the development of natural language processing, the search engine could clarify a user's intent by asking clarifying questions [6] proactively and conversationally. Compared with other approaches, asking clarifying questions is considered a more convenient and satisfying way to obtain a user's ambiguous or faceted search intent [26, 32], and is particularly effective in conversational search systems for limited-bandwidth devices.

Figure 1 illustrates how the search engine asks and how the user responds for clarification. After the user submits a multi-facet query q "headaches", the system asks a question (underlined) and provides several optional items S (i.e., "symptom", "treatment", etc.) to understand the user's intent. If the user clicks an item (such as "treatment"), the query will be refined according to the selection ("headaches treatment" in this case) and search results will be updated. The question is important to improve user experience.

Table 1: Clarifying question templates.

N.	Template
T1	What do you want to know about QUERY?
T2	What do you want to know about this QUERY_DESC?
T3	Which ITEMS_DESC are you looking for?
T4	Who are you shopping for?
T5	What do you want to do with QUERY?
T6	Which QUERY do you mean?

Recently, template-based methods have been applied to clarifying questions generation in conversational search systems. Table 1 shows six common question templates in existing studies [37–39]. The system will scan inversely from T6 to T1 to select a template. Among them, **T6**, **T5**, and **T4** are three separately-treated cases for ambiguous queries, users' different actions, and queries for shopping respectively. When these cases do not occur, **T3** and **T2** will be used when the descriptions for the items (DESC_ITEMS) and the queries (DESC_QUERY) can be found respectively. As templates are pre-defined manually, the key to asking high-quality questions based on template T2 or T3 is to generate a *good description* for a *query* or its *items*. For instance in Figure 1, "medical condition" is a good description of query "headaches", and "type of headaches" also greatly describes items "[migraine, tension, cluster, hormone]". If these descriptions cannot be found, the system will show a *generic questions* like **T1** which reduces user experience [39]. Due to the diversity of web queries and their items, it is challenging to find good descriptions for them in IR systems. Therefore, it brings a large number of generic questions that are not satisfying enough.

In this situation, generating good descriptions is vital for improving the quality and satisfaction of clarifying questions. Existing methods for generating descriptions for queries and items often rely on a static knowledge base. Zamani et al. [37] collected "IsA"-like entity type relations from web texts to form a database. Given a query q and its items S, they used q and S to find the common type from this database and used this type as the description. Similar results can also be generated using well known knowledge bases like WebIsA [29], ConceptGraph [35, 36], and Wikidata [31]. However, the knowledge-based approach has two limitations. **First**, these knowledge bases only contain entities and their descriptions, but many user queries and items are not entities. For example, neither query "google chrome exe" nor any of its items "[32 bit, 64 bit]" is an entity, so descriptions for the query and these two items cannot be found from the knowledge bases. For this case, the system can only ask a *generic question*, like "Select one to refine your search", which gives almost no information and no sense of intelligence to users. **Second**, it does not consider the impact of query q while generating descriptions for items S, and this may make the descriptions for items *less specific*. For example, for query "Olympics" and items "[Beijing, Tokyo, Paris]", existing methods are easy to generate "city" as a description of items. However, "host city" is more specific and brings a better user experience to be exhibited.

To overcome these two limitations, we propose retrieving the top search results of the query to help find better descriptions because we deem that top results contain abundant and relevant contextual information. To prove this, we make statistical analysis in Section 3, showing that top results contain more descriptions of queries and items compared with existing "IsA"-like knowledge bases. According to the statistics, we design three Search Result-based Question Generation (SRQG) algorithms. Specifically, we **first** design a rule-based algorithm **SRQG-Rule** which extracts description candidates from top search results and ranks them by various human-designed features. We **then** train an ensemble learning-to-rank algorithm **SRQG-LTR** to integrate various machine learning models to rank the candidates automatically according to their features. **Furthermore**, we use data generated by the previous two methods as weak supervision signals to train a neural generative model **SRQG-Gen** and generate questions in an end-to-end way.

We evaluate our proposed methods based on the MIMICS [38] dataset. As descriptions of queries or items are incomplete in this dataset, we further employ three annotators who understand our task well to judge the quality of generated **descriptions** and **questions** in a pool-based manner. Experimental results show that our proposed models significantly outperform existing methods in terms of the quality of generated descriptions and questions. SRQG-Rule consistently generates better questions for most queries compared with existing methods, which confirms the usefulness of top results. SRQG-LTR can further improve the generalization ability of SRQG-Rule and generate descriptions and questions with higher quality. SRQG-Gen using top results as input outperforms baseline models with the same structure using "IsA"-like relations as input. The experimental results demonstrate the effectiveness of top results for search clarification. In order to further confirm our conclusion, we make an ablation study to show the importance of each feature used in SRQG-Rule. We also make a case study to intuitively compare the results generated by different algorithms.

In our study, we *do not* research on obtaining items corresponding to a query. We assume that items are given for a query and we focus on generating *questions* with higher quality. We *do not* train a large-scale pre-trained model because we focus on comparing the effectiveness of top search results and knowledge bases for generating clarifying questions. For the coverage of our proposed methods, they are suitable for ambiguous or multi-faceted queries from all topic types whose user intents are usually difficult to understand. On the other hand, they are especially suitable for *transactional* and *informational* queries in web search [15] because items of a query are usually potential transactional or informational user intents.

2 RELATED WORK

Asking Open-domain Clarifying Questions. Aliannejadi et al. first proposed asking clarifying questions in conversational search [5, 6, 13, 20]. They focus on *selecting* questions from a closed set instead of *generating* them. However, queries in IR are complex and diverse, making selecting cannot satisfy users' needs. Zamani et al. first emphasized the importance of clarification in IR [39]. They proposed MIMICS [38] dataset, which includes three subsets: MIMICS-Click, -Explore, and -Manual. Each data in MIMICS consists of a query, several items, and a question. In this paper, other than exploring a better method for generating items, we focus on improving the quality of the "*question*" displayed to users. In existing work, Zamani et al. [37] proposed RTC to collect "IsA"-like relations from web texts, then match descriptions based on

extracted relations to generate questions. They also proposed a Seq2Seq model QLM for generating questions using data generated by RTC as weak-supervised signals. However, both RTC and QLM use a knowledge base as the only information source, so they tend to generate numerous generic questions. Wang and Li [33] applied a multi-task learning framework to template selecting and slot filling jointly. However, all of these works do not consider the completeness and specificity of existing descriptions.

Asking Close-domain Clarifying Questions. Close-domain search clarification is often deemed easier to be implemented than open-domain scenarios because its topics focus on limited areas and it has more available data. Braslavski et al. [7] analyzed a large number of questions and identified common patterns of questions, proving that templates are important for clarifying questions. Rao and Daumé [27] trained a neural network to rank clarifying questions. After that, they tried to generate questions [28] by a generative adversarial network, inspiring us that generative models can improve the generalization of clarifying questions.

Other Relevant Studies. There are some other studies that are related to search clarification. For example, query suggestion utilizes a user's current query to recommend relevant queries [2, 3, 14, 23]. For search clarification, query suggestion is a feasible way to obtain optional items S for query q [37]. Besides, search result diversification [17, 21, 25, 34] and query facets mining [10–12, 16, 18, 19] are also effective ways to clarify user intent, and they can replace query suggestion to obtain corresponding items S for query q. Different from these relevant studies focusing on **finding aspect items**, we aim to **improve the informativeness and readability of clarifying questions to improve user experience**, assuming that the items corresponding to a query are given.

3 ANALYSIS OF CLARIFYING QUESTIONS

In this section, we analyze the composition of a clarifying question, and then carry out two statistical analyses explaining the reason for using top search results to generate clarifying questions.

Generating fluent and informative clarifying questions is challenging due to the lack of data. Recently, Zamani et al. [37] defined several question templates shown in Table 1, like "What do you want to know about this QUERY_DESC?" and "Which ITEMS_DESC are you looking for?". The most important part of a question is the **description** of query q or items S, namely "QUERY_DESC" and "ITEMS_DESC". One way to obtain the descriptions is to build a static offline knowledge base of "IsA"-like entity type relations [37], then match descriptions of queries and items from the knowledge base. For example, given a piece of text "earth *is a* planet", then "planet" is a description of "earth". However, we deem that this kind of knowledge base can just cover descriptions for *entity-like queries or items*, and only a limited number of queries can match this kind of description. For a large range of queries, the system may fail to find a proper description and hence use a generic pre-defined question like "What do you want to know about QUERY?" or "Select one to refine your search" instead.

To alleviate this problem, we propose using the top search results of the query as a complementary data source and finding good descriptions from the results. We argue that top results contain rich relevant and contextualized information which are superior to static "IsA"-like knowledge bases. To prove the effectiveness of top results, we make a statistical analysis to survey whether top results can cover more queries, items, and their descriptions compared with existing "IsA"-like knowledge bases. To achieve this, we first sample 400, 400, and 200 non-generic clarifying questions from three subsets of MIMICS [38] mentioned in Section 2 respectively. In each part of the data, half of the questions contain a description of the query, and the other half of the questions contain a description of items. These descriptions are extracted from "IsA"-like knowledge bases using existing methods [37]. We extract descriptions from these questions and find them in the top results. Specifically, we find descriptions of query $d(q)$ in top results of query q, and find descriptions of items $d(S)$ in top results of the concatenation of q and S, recorded as (q, S). For example, suppose q is "headaches" and S is "[symptom, treatment, causes, diagnosis]", then (q, S) is "headaches symptom treatment causes diagnosis". We design this combined query to better obtain descriptions of items S since top results can integrate more co-occurrence information of q and S. Table 2 shows the statistical results. In the first column, q top x means top x result pages of q, and (q, S) top x means top x result pages of (q, S). Each value indicates the rate of $d(q)$ or $d(S)$ that can be found from top x results of q or (q, S).

Table 2: Statistics on MIMICS dataset: proportion of descriptions extracted from existing "IsA"-like database that can be found in top search results.

Dataset	Click		ClickExplore		Manual	
Description	$d(q)$	$d(S)$	$d(q)$	$d(S)$	$d(q)$	$d(S)$
q top 5	0.580	0.255	0.675	0.205	0.480	0.200
q top 10	0.755	0.345	0.760	0.300	0.600	0.250
q top 20	0.975	0.570	0.950	0.645	0.910	0.420
q top 50	**1.000**	0.740	**1.000**	0.790	**1.000**	0.730
(q, S) top 5	0.535	0.345	0.450	0.310	0.420	0.310
(q, S) top 10	0.685	0.450	0.705	0.465	0.540	0.340
(q, S) top 20	0.895	0.720	0.870	0.715	0.810	0.550
(q, S) top 50	1.000	**0.860**	0.990	**0.870**	0.920	**0.750**

Table 3: Statistics on MIMICS dataset: proportion comparison of query and items that can be found from top search results, WebIsA and Concept Graph (CG).

Bing Top Search Results					
Proportion	$o(q)$	$o(S)$50%	$o(S)$70%	$o(S)$80%	$o(S)$100%
q top 5	0.657	0.504	0.445	0.440	0.270
q top 10	0.726	0.551	0.512	0.503	0.295
q top 20	0.782	0.615	0.559	0.534	0.341
q top 50	**0.843**	0.722	0.613	0.606	0.362
(q, S) top 5	0.548	0.683	0.599	0.573	0.446
(q, S) top 10	0.586	0.751	0.646	0.624	0.505
(q, S) top 20	0.653	0.812	0.732	0.688	0.558
(q, S) top 50	0.691	**0.850**	**0.793**	**0.750**	**0.631**
Existing "IsA"-like Knowledge Bases					
Proportion	$o(q)$	$o(S)$50%	$o(S)$70%	$o(S)$80%	$o(S)$100%
WebIsA	0.301	0.537	0.424	0.390	0.322
CG	0.263	0.546	0.437	0.378	0.331

Table 2 shows that, with the increasing number of returned top search results, more descriptions extracted from "IsA"-like relations can be found. In three subsets of MIMICS, we can find **descriptions of all queries** (bold in the fourth line) and **about 85% descriptions of items** (bold in the eighth line) averagely within top 50 results. Besides, if only using *original query* q to submit (the upper four lines), we can find **more descriptions of query** in top results. If we concatenate query and items as a *combined query* (q, S) (the lower four lines) to submit, we can find **more descriptions of items**. It indicates that the combined query (q, S) can obtain abundant contextual information of the query q and its corresponding items S, thereby more descriptions of S can be found.

To further illustrate the advantage of top search results over "IsA"-like data resources, we then compare the proportion of queries and items occurring in top results with existing "IsA"-like databases. We randomly sample 1,000 data from MIMICS and find these queries and items from top results, WebIsA knowledge base, and Concept Graph for comparison. Table 3 shows the results. In this table, $o(q)$ means the percentage of query q that can be found, and $o(S)x\%$ means the percentage of $x\%$ of items that can be found. It illustrates that **compared with the "IsA"-like knowledge bases, top results contain more queries and items**, thus it has a higher probability to find descriptions. Similar to Table 2, Table 3 also proves that, **proportion of query will be higher in top results of q, and proportion of items will be higher in top results of (q, S)**.

4 METHODS

Based on the statistics in Section 3, we aim to use top results of q to extract descriptions of the query, and use top results of (q, S) to extract descriptions of the items. We propose a framework composed of **three Search Result-based Question Generation algorithms (SRQG-Rule, SRQG-LTR, and SRQG-Gen)** shown in Figure 2. In SRQG-Rule, we first retrieve top-n results of q and (q, S) respectively, then extract all plain texts and list structures from the results, which contain important information in HTML. After that, we use a rule-based extractor to get a set of description candidates (luxury watch, man, brand, etc. in Figure 2) from extracted texts and lists, and further score and rank all these candidates based on various human-designed features. Since a candidate c can describe a query q or its items S, we select two candidates with the highest score for query and items as c_q and c_S from the results of q and (q, S) respectively, and then choose the one with a higher score to combine with template T2 or T3 shown in Table 1 to form a question. We further label the ranked list for training a ranking model SRQG-LTR and use data generated by the above two algorithms as weak supervision signals to train a generative model SRQG-Gen.

4.1 SRQG-Rule

SRQG-Rule aims to obtain description candidates for the query and items from the extracted texts and lists in top search results, then calculate the value of various human-designed features for each candidate to select optimal candidate(s). It is composed of *Lists and Texts Extractor*, *Candidates Extractor* and *Candidates Ranker*.

4.1.1 Lists and Texts Extractor. In one HTML document of the search results, important contents usually exist in plain texts and list structures [10]. Plain texts are long paragraphs that contain

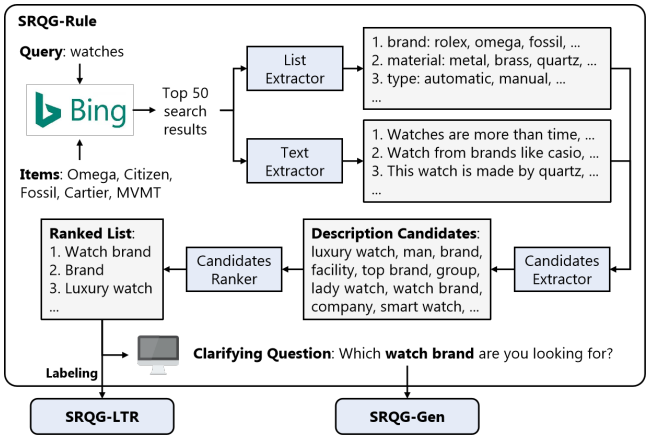

Figure 2: An overview of our proposed methods with an example of finding descriptions for items.

unstructured natural language information. List structures that frequently appear in HTML usually illustrate parallel information, like attributes of goods on e-commerce websites, and tables on encyclopedia websites. These two resources are significant to obtain most of the information from top results. Therefore, we first extract HTML paragraphs with more than 60 characters as text set $T = [T_i, 1 \leqslant i \leqslant |T|]$. We further implement an effective and efficient algorithm [10] to extract parallel structures from HTML tags, plain texts, and repeat regions. We simultaneously extract contents from the context of each extracted parallel structure, because the context may include a description of the parallel structure. We combine parallel structures with their contexts as list set $L = [(L_i^C, L_i^S), 1 \leqslant i \leqslant |L|]$, where L_i^S denotes a parallel structure, and L_i^C denotes the corresponding context of list L_i^S.

4.1.2 Candidates Extractor. It is challenging to directly extract descriptions from texts or lists. Therefore, we aim to use some rules to build a candidate set of descriptions, then we just need to find the best candidate in this set. Based on our observations, for a query q, its descriptions are highly possible to appear in "IsA"-like relations. Thus we first use 59 templates [29] to obtain "IsA"-like patterns including hyponyms and hypernyms from top search results, then obtain nouns and noun phrases in hypernyms as query description candidates. For example, for the text "supernoobs is a Canadian flash animated series produced by ...", we can obtain "series" and "Canadian flash animated series", as description candidates of the query "supernoobs". For items S, their descriptions are highly possible to appear in the list contexts and the neighbors of S in plain texts, so we extract nouns and noun phrases from these parts as items description candidates. We apply CoreNLP [22] part-of-speech module to extract nouns and constituency parsing module to extract noun phrases. Besides, we also extract candidate descriptions from WebIsA [29] and Concept Graph [35, 36] as supplement. Using the above rules, we obtain an average of 34.4 description candidates for query q and 41.7 description candidates for items S.

4.1.3 Candidates Ranker. To rank candidates based on comprehensive considerations like statistical information and semantic

Table 4: Notations in our approaches.

N.	Explanation	N.	Explanation
q	Query	C_q	Candidate set of q
S	Items	C_S	Candidate set of S
c	Candidate	τ	Threshold
k	Adjustment coefficient	p	Importance coefficient
L_i	Extracted ith list	T_i	Extracted ith text

similarity etc., we design various features for each candidate c, including list statistical feature f_L, text statistical features f_T, semantic feature f_S, entity feature f_E and inhibition feature f_I. Each feature scores a candidate from a sole and different perspective. All features are added together as the total score of a candidate c:

$$\text{score}(c) = f_L(c) + f_T(c) + f_S(c) + f_E(c) + f_I(c). \tag{1}$$

We rank all description candidates based on their scores and select two candidates with the highest score for query q and items S as c_q and c_S respectively. To determine whether to return a description for query or for items, we set two thresholds τ_q and τ_S. If $\text{score}(c_S) \geq \max(\text{score}(c_q), \tau_S)$, we set c_S as items description. Else, if $\text{score}(c_q) \geq \max(\text{score}(c_S), \tau_q)$, we set c_q as query description. Else, candidates ranker return *None*, ensuring that when candidate scores are all low and candidates are likely to be wrong, the system can generate a generic question instead of a wrong question.

For the rest of this section, we will introduce features used by SRQG-Rule one by one. All notations in equations are shown in Table 4. For each feature, we manually set an adjustment coefficient k and an importance coefficient p to determine the importance of each feature, and use the tanh function for feature normalization by controlling the value range of each feature. For each feature f and each coefficient k and p, we use subscript to indicate the feature name and use superscript to indicate whether the feature is calculated for a query or its corresponding items.

1. List Statistical Feature f_L. A list L_i extracted from HTML can be formalized as (L_i^C, L_i^S), where L_i^S is parallel structure and L_i^C is list context which may include description for L_i^S. The structure is corresponding to aspect items S and their description $d(S)$. Therefore, list structures are naturally suitable for finding descriptions of items S. Intuitively, the more a candidate c appears in L_i^C, and the more items appear in L_i^S, the higher probability will be that the c is a good description of aspect items S. Thus we have:

$$f_L^q(c) = 0; \quad f_L^S(c) = p_L^S \cdot \tanh(k_L^S \sum_i I(c, L_i^C) \cdot \frac{|S \cap L_i^S|}{|S|}), \tag{2}$$

where $I(a, b)$ is indicator function. If a occurs in b, $I(a, b) = 1$, else $I(a, b) = 0$.

2. Text Statistical Features f_T. Besides list structures, plain texts also contain abundant information which may include good descriptions for q and S. Therefore, we design several text statistical features f_T to capture statistical information of candidates in plain texts. We divide f_T into pattern feature, distance feature, co-occurrence feature, frequency feature, and inclusion feature. We add these five features together as text statistical features f_T:

$$f_T(c) = f_p(c) + f_d(c) + f_c(c) + f_f(c) + f_i(c). \tag{3}$$

- Pattern Feature f_p. Patterns are important to extract descriptions explicitly from plain texts. For example, in the sentence "programming language *such as* Java, Python, and C", "programming language" is the hypernym, "java", "Python" and "C" are hyponyms. The hypernym is a good description for hyponyms. For query q, if a candidate c appears in hypernym and q appears in hyponyms, c has a high probability to be a description of q, thus we assign a higher score to c for q. For items S, if a candidate c appears in hypernym and more items appear in hyponyms, c has a higher probability to be a description of S, thus we assign a higher score to c for S. In summary, we calculate f_p as follows:

$$\begin{aligned} f_p^q(c) &= p_p^q \cdot \tanh(k_p^q \sum_i I(c, P_i^H) \cdot I(q, P_i^T)), \\ f_p^S(c) &= p_p^S \cdot \tanh(k_p^S \sum_i I(c, P_i^H) \cdot \frac{|S \cap P_i^T|}{|S|}), \end{aligned} \tag{4}$$

where P^H is the set of *hypernyms* and P^T is the set of *hyponyms* corresponding to P^H.

- Distance Feature f_d. Intuitively, the closer candidate c is to query q or items S, the more likely it is to be a good description of the query or the items. Therefore, we set a distance function to calculate distance between candidate c and items or query, then calculate distance feature f_d based on this distance function:

$$\begin{aligned} f_d^q(c) &= p_d^q \cdot \tanh(k_d \sum_i d(c, S, T_i)), \\ f_d^S(c) &= p_d^S \cdot \tanh(k_d \sum_i d(c, q, T_i)). \end{aligned} \tag{5}$$

Here $d()$ is defined as reciprocal of the sum of distances:

$$d(c, q, T_i) = \sum_j \frac{1}{\text{dist}(c, q_{ij})}, \qquad d(c, S, T_i) = \sum_j \frac{1}{\text{dist}(c, S_{ij})}, \tag{6}$$

where q_{ij} indicates each query occur in T_i, S_{ij} indicates each item in S occur in T_i. The distance function $\text{dist}(a, b)$ is defined as:

$$\text{dist}(a, b) = \begin{cases} |\text{pos}(a) - \text{pos}(b)| & \text{if } |\text{pos}(a) - \text{pos}(b)| \leqslant 50; \\ +\infty & \text{for else.} \end{cases} \tag{7}$$

Here $\text{pos}(x)$ returns position of x in a paragraph. If the distance between a and b is longer than 50 characters, we deem there is no strong connection between them anymore.

- Co-occurrence Feature f_c. We also capture co-occurrence information between candidate c and q or S. Naturally, candidate c appears in the same sentence with query q or items S are more likely to be their good description. Therefore, the co-occurrence feature f_c is defined as follows:

$$\begin{aligned} f_c^q(c) &= p_c^q \cdot \tanh(k_c^q \sum_i \sum_j \text{occ}(c, q, T_{ij})), \\ f_c^S(c) &= p_c^S \cdot \tanh(k_c^S \sum_i \sum_j \sum_l \text{occ}(c, S_l, T_{ij})). \end{aligned} \tag{8}$$

For function $\text{occ}(a, b, c)$, if a and b occur in c simultaneously, $\text{occ}(a, b, c) = 1$, else $\text{occ}(a, b, c) = 0$. T_{ij} indicates the jth sentence in T_i, S_l denotes the lth item in S.

- Frequency Feature f_f. Phrases with higher frequency in texts are likely to have a strong correlation with a query and items. Therefore, we count candidate c in top results as frequency feature:

$$f_f^q(c) = p_f^q \cdot \tanh(k_f^q \sum_i N_{T_i}(c)),$$
$$f_f^S(c) = p_f^S \cdot \tanh(k_f^S \sum_i N_{T_i}(c)), \tag{9}$$

where $N_{T_i}(c)$ indicates the frequency that c occurs in text T_i.

- Inclusion Feature f_i. If a candidate c contains other candidate(s) as sub-string(s), it is likely to be a more specific description. For example, as a description of items "[64 bit, 32 bit]", *version* is not as specific as *bit version* or *bit version of google chrome*. Therefore, we need to increase score of these specific descriptions. For candidate c, we calculate inclusion feature f_i as follows:

$$f_i^q(c) = p_i^q \cdot \tanh(k_i^q \sum_j I(C_{j\neq c_{arg}}, c) \times \text{score}'(C_{j\neq c_{arg}})),$$
$$f_i^S(c) = p_i^S \cdot \tanh(k_i^S \sum_j I(C_{j\neq c_{arg}}, c) \times \text{score}'(C_{j\neq c_{arg}})), \tag{10}$$

where C_j is the jth candidate in candidate set C, c_{arg} is the index of candidate c. $\text{score}'(C_{j\neq c_{arg}})$ indicates scores of other candidate without c, therefore, including candidates with higher score in other features will increase value of f_i of candidate c.

3. Semantic Feature f_S. The list statistical feature and text statistical features can only obtain statistical or lexical information from HTML files. To improve algorithm performance, we introduce semantic feature f_S to integrate semantic information. In detail, we calculate semantic similarity between candidate c and query, or between c and items as semantic features. We apply BERT [9], a pre-trained language model, to encode candidate c, query q, and each item S_i. In order to fuse contextual information, we obtain context $\mathcal{T}$ of candidate c, query q, and each item S_i from top search results as $\mathcal{T}_c$, $\mathcal{T}_q$, and $\mathcal{T}_{S_i}$ respectively. We first calculate representations of c, q, and S by BERT as: $R_c = \frac{1}{l_c}\sum_{j=1}^{l_c} \text{BERT}(c_j, \mathcal{T}_c)$, $R_q = \frac{1}{l_q}\sum_{j=1}^{l_q} \text{BERT}(q_j, \mathcal{T}_q)$, and $R_{S_i} = \frac{1}{l_{S_i}}\sum_{j=1}^{l_{S_i}} \text{BERT}(S_{ij}, \mathcal{T}_{S_i})$. We then calculate cosine similarity between representations of c and q, S encoded by BERT as semantic feature f_S:

$$f_S^q(c) = p_S^q \cdot \tanh(k_S^q \cdot \text{cosine}(R_c, R_q),)$$
$$f_S^S(c) = p_S^S \cdot \tanh(k_S^S \cdot \frac{1}{l_S}\sum_i \text{cosine}(R_c, R_{S_i})). \tag{11}$$

4. Entity Feature f_E. For ensuring that descriptions of entities can be found, we use WebIsA [29] and Concept Graph [35, 36] as supplementary resources to find descriptions of entities:

$$f_E(c) = p_E \cdot \tanh(k_E(N_w(c) + N_c(c))), \tag{12}$$

where $N_w(c)$ and $N_c(c)$ are frequencies of c that occurs in WebIsA and Concept Graph respectively.

5. Inhibition Feature f_I. Some items sets corresponding to some queries have small consistency and large coverage [39], so it is not necessary to generate a clarifying question for them. For example, the items corresponding to the query "headaches" are "[symptom, treatment, diagnosis, causes]" shown in Figure 1. In this circumstance, even humans cannot make a good description of these items, thereby candidates extracted from top search results are probably wrong. We aim to reduce errors in this case. Therefore, we want to reduce scores for this kind of items. Specifically, we first calculate the average semantic similarity between each item as s_S using BERT, then define feature f_I as:

$$f_I^S = \begin{cases} 0 & \text{if } s_S \geqslant \tau_{is}, \\ -p_I(\tau_{is} - s_S) & \text{if } s_S < \tau_{is}. \end{cases} \tag{13}$$

Where τ_{is} is a manually set threshold. We apply the inhibition feature f_I to all description candidates.

Algorithm 1: SRQG-Rule

Input: Query q, Items Set S
Output: Clarifying Question Q

```
1  Judge if query q is an ambiguous query or a faceted query;
2  For ambiguous query, use template T6;
3  For faceted query:
4  if items set S are all actions:
5  |  use template T5.
6  else if S are related to people and q is related to a product:
7  |  use template T4.
8  else if candidates extractor and ranker returns a description:
9  |  use template T3 or T2.
10 else if ITEMS_DESC can be found from "IsA"-like database:
11 |  use template T3.
12 else if QUERY_DESC can be found from "IsA"-like database:
13 |  use template T2.
14 else
15 |  use template T1.
```

4.1.4 Algorithm Flow. Since SRQG-Rule is a rule-based algorithm depending on templates, we first gather existing clarifying question templates [37, 39] to get a unified set of templates shown in Table 1. Among them, template T6 is for ambiguous queries, T5 is for cases where items are all actions, and T4 is used when the query is a commodity and items are related to people. The above three are special and frequent cases in MIMICS dataset, so they are handled separately by rules. T3 and T2 describe items and queries respectively which we focus on, while T1 is a pre-defined generic clarifying question that is equivalent to "Select one to refine your search". The algorithm flow of SRQG-Rule is shown in Algorithm 1. Among them, lines 8 and 9 are our improvements that leverage top results compared with the previous rule-based algorithm RTC [37].

4.2 SRQG-LTR

SRQG-Rule ranks description candidates based on human-designed rules and thresholds, which could be sub-optimal in some cases. Manually tuning the weights of features is also time-consuming. We hope the weights of features can be determined automatically, so the system can select good descriptions more effectively and efficiently. Therefore, we further design SRQG-LTR, an ensemble learning-to-rank model, to **automatically rank candidates** according to the extracted features. Formally, we aim to build a pairwise learning-to-rank model $f(y|x_1, x_2)$. The input vector x_1 and x_2 are features of

two description candidates and the output y is a class label denoting whether x_1 should be ranked higher than x_2. To build a training set for SRQG-LTR, we randomly sample 500 pieces of data of query and corresponding items from MIMICS, run SRQG-Rule to get description candidates, then ask trained annotators to label some of these candidates as supervised data. The labeling criteria will be declared in Section 5.1. We expect the model to **learn patterns from the training data to automatically weight features and improve the quality of clarifying questions by generalizing SRQG-Rule**. Besides features mentioned in Section 4.1, we also add the length and word number of all queries, items, and description candidates as supplementary features.

4.3 SRQG-Gen

To further demonstrate the effectiveness of using *top search results* in generating clarifying questions, we devise a Seq2Seq generative model SRQG-Gen to generalize SRQG-Rule and SRQG-LTR to improve the quality of clarifying questions. To achieve this, the structure of SRQG-Gen remains basically consistent with QLM [37], an encoder-decoder network, which receives a query, several items, and their corresponding descriptions as inputs, and outputs a generated clarifying question. Based on QLM as a baseline model, we replace the description of a query or items extracted from "IsA"-like databases with that extracted from top results as weak-supervision training data. Our purpose is to prove that, with the same network structure with QLM, **descriptions of query and items obtained from top search results are better than those obtained from "IsA"-like database, so they can better train a neural model as weak supervision signals**. Therefore, we *do not* focus on designing novel network structures. The network can be simply replaced by some other structures like convolutional network [4], transformer [30], or pre-training based models [9].

5 EXPERIMENTS

5.1 Human Labeling for Evaluation

We apply human annotation to evaluate generated descriptions and clarifying questions. Evaluation metrics, annotation process, and amount of evaluation data will be basically consistent with the existing work [37–39] as far as possible. Specifically, we hire three annotators with compensation who have a Master's degree, then explain our task explicitly until they totally understand and are able to restate the task completely. An online meeting with a host will be held for detailed communication. The three annotators choose a score form {0, 1, 2, 3} according to the criteria of descriptions or questions shown in Table 5. The criteria focus on both **correctness** and **specificity** of **descriptions and clarifying questions**. For each piece of data, each annotator selects a score individually and the final score is based on a **majority vote** among three annotators. If three annotations for one piece of data are totally different from each other (like 0, 1, and 2), the three annotators launch a quick online meeting and discuss with each other in turn to determine the final score, then it will be recorded by the host. This situation of lacking agreement accounts for less than 5% of all annotations, indicating that the three annotators have a high degree of agreement. The overall Fleiss' kappa among three annotators is 71.36%, which also means that the three annotators annotate consistently.

5.2 Implementation Details

For SRQG-Rule, we obtain top results retrieved by Bing Search API v7 [1]. For each query, we get top 50 results for the original query q and the combined query (q, S) respectively. Illegal and unsafe websites will be deleted manually by some keywords. To determine the optimal combination of parameters in SRQG-Rule, we implement all features, then empirically tune thresholds τ, adjustment coefficient k, and importance coefficient p of each feature mentioned in Section 4.1.3 by grid search with a step of 0.1 in the range of (0, 1].

We use Scikit-Learn [8, 24] to implement SRQG-LTR by the ensemble of six machine learning models, including K-Nearest Neighbors, SVM with a Gaussian kernel, Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting Decision Tree. All of these models use *default parameters* in Scikit-Learn. They are trained by *pairwise* approach and are then tested on evaluation data based on the majority vote. We apply TensorFlow 2.0 [1] to implement the network structures of QLM (baseline) and SRQG-Gen. Since we design SRQG-Gen to further illustrate that descriptions extracted from top results are of higher quality than that extracted from "IsA"-like databases, all parameters of SRQG-Gen, including the network structure, optimizer, learning rate, batch size, division of training data and validation data, are totally consistent with the implementation of QLM [37] for comparison[2].

5.3 Data

We fail to obtain the "IsA"-like knowledge base used in the existing study [37] because of its inaccessibility. Alternatively, we use the WebIsA [29] and ConceptGraph [35, 36] knowledge bases which are similar and competitive in structure and scale with [37] for obtaining descriptions of entities. As for training data, we randomly sample 500 pieces of data from MIMICS for training SRQG-LTR, and 100k pieces of data from MIMICS[3] for training QLM and SRQG-Gen. We ask three annotators to manually label data for SRQG-LTR according to criteria in Section 5.1, resulting in about 200k pieces of pairwise data to train two ranking models for ranking descriptions of queries and items respectively. Following [37], we first reproduce RTC to generate weak supervision data, then use these data to train QLM. Similarly, in our proposed methods, we first run SRQG-Rule to get an original ranked list of description candidates, then apply SRQG-LTR to refine the list, and finally select the top-1 description to generate a clarifying question. We use all these questions as weak supervision data to train SRQG-Gen.

Since user queries are complex and diverse, their corresponding items can be derived from different data resources. We consider the following three datasets for evaluation, representing three different situations: (1) Sampling 100 queries with "overall good" label extracted from **MIMICS-Manual**: MIMICS-Manual is a part of MIMICS containing manual labels. In this situation, most queries are *head* queries sampled from query logs, and their items are of high consistency. (2) Sampling 100 queries from **MIMICS-Click**: MIMICS-Click contains many *tail* queries randomly sampled from query logs, and their corresponding items sometimes have little consistency, making it more difficult to find descriptions. (3) Sampling

[1] Bing Search API: https://www.microsoft.com/en-us/bing/apis/bing-web-search-api
[2] Implementation details are available at https://github.com/zillion-zhao/SRQG
[3] MIMICS Dataset: https://github.com/microsoft/MIMICS

Table 5: Criteria of scoring clarifying questions and descriptions for a query and its corresponding items.

Score	Criteria
0	Wrong or illogical clarifying question. Question is irrelevant to query and items, or description of query or items is wrong.
1	Generic clarifying question like "What do you want to know about QUERY?" and "Select one to refine your search."
2	Correct clarifying question whose description for query or items is not specific enough, but still acceptable to users.
3	Correct and specific clarifying question which can give users a sense of intelligence and provide better user experience.

Examples

Query: Olympics, **Items**: [Beijing, Tokyo, Paris, Rome]		**Query**: google chrome exe, **Items**: [32 bit, 64 bit]	
0	Which **software** are you looking for?	0	Which **information** are you looking for?
1	What do you want to know about **Olympics**?	1	What do you want to know about **google chrome exe**?
2	Which **city** are you looking for?	2	Which **version** are you looking for?
3	Which **host city** are you looking for?	3	Which **bit version of chrome** are you looking for?

Table 6: Evaluation results for query and items descriptions, description candidate lists and clarifying questions. ‡ and † denotes significant improvement compared with the best baseline with p-value < 0.01 and p-value < 0.05 respectively.

Evaluation	Query and Items Descriptions						Description Candidate List			Clarifying Questions					
Overall Good	3	2	1	0	linear	exp	NDCG@3	MRR	P@1	3	2	1	0	linear	exp
RTC	0.31	0.19	**0.34**	**0.16**	1.65	3.08	0.468	1.372	0.50	0.35	0.17	0.30	**0.18**	1.69	3.26
QLM	-	-	-	-	-	-	-	-	-	0.32	0.24	**0.31**	0.13	1.75	3.27
SRQG-Rule	**0.57**	0.21	0.07	0.15	2.20‡	4.69‡	0.725‡	2.102‡	0.78‡	**0.61**	0.18	0.07	0.14	2.26‡	4.88‡
SRQG-LTR	0.55	**0.29**	0.05	0.11	**2.28**‡	**4.77**‡	**0.774**‡	**2.499**‡	**0.84**‡	0.59	**0.27**	0.04	0.12	**2.35**‡	**4.98**‡
SRQG-Gen	-	-	-	-	-	-	-	-	-	0.60	0.18	0.07	0.15	2.23‡	4.81‡
Query Log	3	2	1	0	linear	exp	NDCG@3	MRR	P@1	3	2	1	0	linear	exp
RTC	0.25	0.06	**0.62**	0.06	1.49	2.55	0.302	0.778	0.31	0.27	0.06	**0.59**	0.07	1.52	2.66
QLM	-	-	-	-	-	-	-	-	-	0.22	0.15	0.52	0.11	1.48	2.51
SRQG-Rule	**0.46**	0.07	0.28	**0.19**	1.80†	3.71†	0.584‡	1.755‡	0.53†	0.50	0.07	0.24	**0.19**	1.88†	3.95†
SRQG-LTR	0.45	**0.15**	0.29	0.11	**1.94**‡	**3.89**‡	**0.601**‡	**1.835**‡	**0.60**‡	0.48	**0.15**	0.27	0.10	2.01‡	4.08‡
SRQG-Gen	-	-	-	-	-	-	-	-	-	**0.51**	0.14	0.23	0.12	**2.04**‡	**4.22**‡
Q-Dim	3	2	1	0	linear	exp	NDCG@3	MRR	P@1	3	2	1	0	linear	exp
RTC	0.15	0.25	**0.52**	0.08	1.47	2.32	0.362	0.949	0.40	0.15	0.25	**0.52**	0.08	1.47	2.32
QLM	-	-	-	-	-	-	-	-	-	0.11	**0.27**	0.51	0.11	1.38	2.09
SRQG-Rule	0.41	0.20	0.25	**0.14**	1.88†	3.72†	0.598†	1.673†	0.61‡	0.41	0.20	0.25	0.14	1.88†	3.72†
SRQG-LTR	**0.43**	**0.25**	0.23	0.09	**2.02**‡	**3.99**‡	**0.617**‡	**1.669**†	**0.68**‡	**0.43**	0.25	0.23	0.09	**2.02**‡	**3.99**‡
SRQG-Gen	-	-	-	-	-	-	-	-	-	0.38	0.21	0.22	**0.20**	1.78	3.51

100 queries where items S are extracted from top results of query q: In the first two cases, items are obtained from query logs. We hope that when the items come from different sources, our proposed methods can also generate good clarifying questions. Therefore, we randomly sample data from **UserQ and RandQ** [10], two publicly available datasets for finding query dimensions. In these two datasets, items are extracted from the query's top results. For the evaluation dataset scale, we try to stay consistent with previous work to ensure the preciseness of the experiment. All evaluation results are presented in Table 6. We briefly denote the results on three datasets as **Overall Good**, **Query Log** and **Q-Dim** respectively.

5.4 Evaluating Descriptions of Query and Items

Since our approaches focus on finding descriptions for query q and items S, we first evaluate generated descriptions independent of questions. As each generated description is scored by a 4-level scale (0, 1, 2, 3) shown in Table 5, we report the distribution of the scores and calculate a linear score and an exponential score proposed by [37] for each algorithm. The results are shown in the left part of Table 6. Since QLM and SRQG-Gen are generative models and cannot generate independent descriptions, we only report the results for RTC, SRQG-Rule, and SRQG-LTR.

Overall in all three situations, SRQG-Rule outperforms RTC significantly, and SRQG-LTR further improves the overall quality of descriptions. We observe that SRQG-LTR achieves the highest linear score and exponential score in all three situations. SRQG-Rule can generate more descriptions scored by 3 or 2 and fewer descriptions scored by 1 (generic descriptions). It indicates that top search results contain more appropriate and specific descriptions for a query and items. The number of 3-score results generated by SRQG-LTR is about the same as that generated by SRQG-Rule, but SRQG-LTR has an obvious improvement in the number of 2-score results. However, in "Query Log" and "Q-Dim" evaluation datasets, SRQG-Rule is prone to generate more wrong or illogical descriptions scored by 0, with 19% and 14% respectively. It manifests that rule-based methods sometimes extract wrong descriptions from

Table 7: Ablation study for each feature in SRQG-Rule.

Data	Overall Good		Query Log		Q-Dim	
Feature	D-l	NDCG	D-l	NDCG	D-l	NDCG
w/o. f_L	2.13	0.697	1.77	0.571	**1.52**	**0.517**
w/o. f_P	2.18	0.714	1.78	0.571	1.75	0.567
w/o. f_d	2.04	0.633	1.66	0.556	1.68	0.551
w/o. f_c	2.13	0.708	1.78	0.562	1.84	0.584
w/o. f_f	2.07	0.635	1.70	0.558	1.53	0.529
w/o. f_i	2.10	**0.629**	1.75	**0.523**	1.75	0.562
w/o. f_S	2.16	0.717	1.79	0.580	1.82	0.577
w/o. f_E	**1.99**	0.648	**1.61**	0.547	1.60	0.535
w/o. f_I	2.18	0.722	1.77	0.581	1.67	0.580
S-Rule	2.20	0.725	1.80	0.584	1.88	0.598

top search results. On the other hand, SRQG-LTR can re-rank the description candidate lists returned by SRQG-Rule and reduce the proportions of 0-score descriptions in "Query Log" and "Q-Dim" to 11% and 9% respectively. This is because, with supervised learning, the model can automatically learn the weight of each feature, and thereby many candidates that fall behind in the ranking of SRQG-Rule can be improved in the ranking of SRQG-LTR.

5.5 Evaluating Description Candidate Lists

Usually, a query or several items can be well described by multiple descriptions. For instance, items "[Windows 7, Windows 10, Windows xp]" can be described as "*operating system*" or "*version of Windows*". Multiple descriptions can also improve the diversity of questions, for example, the conversational search system can randomly select different descriptions in top-3 candidates in different sessions to improve the diversity of clarifying questions. Therefore, we also evaluate the description candidates list returned by each method, with various ranking metrics including **NDCG@3**, **MRR** and **P@1**. In order to calculate metrics like NDCG and MRR, we select the top-3 candidates returned by each algorithm to form a pool, then compute ranking metrics for each algorithm based on the pooled descriptions. The results are shown in the middle of Table 6. According to the results, SRQG-LTR outperforms SRQG-Rule which performs better than RTC in returned descriptions. The results confirm that top search results not only return a single good description, but can also bring a list of high-quality descriptions where the top-n elements in the list can provide a comprehensive and specific description for queries or items. In three evaluation datasets, SRQG-Rule shows significant improvement in all three metrics. Compared with SRQG-Rule, SRQG-LTR improves NDCG@3 and MRR slightly and improves P@1 by an average of 6.67%.

5.6 Evaluating Clarifying Questions

Our final purpose is to generate high-quality clarifying questions. Therefore, we also apply the criteria in Table 5 to evaluate questions displayed to users. Automatic metrics such as ROUGE and BLEU are poorly correlated with user satisfaction and clarification quality [38], so they are not used for evaluation. The evaluation results are shown on the right side of Table 6. Compared with RTC, SRQG-Rule can consistently generate more high-quality clarifying questions by finding better descriptions from top search results. SRQG-LTR further improves SRQG-Rule by re-ranking returned description candidate list. SRQG-Gen gets the best result in the "Query Log" case, but it does not perform as well as SRQG-Rule and SRQG-LTR in the other two situations. All proposed SRQG models outperform baselines significantly.

Table 8: Four example outputs of different algorithms.

Query	google chrome exe	**Items**	[64 bit, 32 bit]
RTC	What do you want to know about **google chrome exe**?		
QLM	What do you want to know about **google chrome exe**?		
S-Rule	Which **bit version of chrome** are you looking for?		
S-LTR	Which **bit version of google chrome** are you looking for?		
S-Gen	Which **bit version of chrome** are you looking for?		
Query	quiet riot	**Items**	[song, member, album]
RTC	What do you want to know about this **band**?		
QLM	What do you want to know about this **band**?		
S-Rule	What do you want to know about this **heavy metal band**?		
S-LTR	What do you want to know about this **heavy metal band**?		
S-Gen	What do you want to know about this **metal band**?		
Query	facial tingling	**Items**	[left side, right side]
RTC	What do you want to know about **facial tingling**?		
QLM	What do you want to know about this **disease**?		
S-Rule	Which **side** are you looking for?		
S-LTR	Which **side of the body** are you looking for?		
S-Gen	Which **side of your head** are you looking for?		
Query	cabins in asheville nc	**Items**	[for rent, for sale]
RTC	What do you want to know about **cabins in asheville nc**?		
QLM	What do you want to know about this **county**?		
S-Rule	Which **owner** are you looking for?		
S-LTR	Which **single family home** are you looking for?		
S-Gen	Which **owner for home** are you looking for?		

Specifically, the evaluation results of RTC, SRQG-Rule, and SRQG-LTR are similar to *the results of descriptions* but with higher scores in general, because some questions are generated by templates T4, T5, and T6 which do not consider descriptions for some queries, especially for "Overall Good" and "Query Log" situations. For the "Q-Dim" situation, there are no questions generated by T4, T5, and T6, thus results of RTC, SRQG-Rule, and SRQG-LTR for descriptions and questions are identical. For generative models, QLM improves RTC in the "Overall Good" situation slightly, which is consistent with previous study [37]. SRQG-Gen significantly outperforms all baselines and even performs better than SRQG-LTR in "Query Log" data, confirming descriptions extracted from top results are of higher quality than descriptions extracted from "IsA"-like relation knowledge bases. However, both QLM and SRQG-Gen perform badly in "Q-dim" data. This is because training data of QLM and SRQG-Gen are sampled from MIMICS dataset, and items in MIMICS are obtained from query logs, while items in the "Q-dim" data are extracted from the top results of the query. This leads to a different distribution between training data and evaluation data, therefore generative models cannot perform well in this situation. A way to alleviate this problem is to extract *items* from top results to replace items extracted from query logs for generative models.

5.7 Ablation Study

In our proposed three algorithms, one of our main conclusions is that the features used in SRQG-Rule are important to obtain high-quality descriptions of queries and items, therefore SRQG-Rule can

generate better clarifying questions compared with existing rule-based baseline RTC. The descriptions and questions generated by SRQG-Rule can be further utilized in SRQG-LTR and SRQG-Gen to improve clarification quality. In the above conclusion, **features** play an important role. To prove the effectiveness of each feature for extracting high-quality descriptions for queries and items from top search results, we conduct an ablation study by removing one feature at a time to compare the performance. Similar to our evaluation in Table 6, we evaluate the results in three datasets: Overall Good, Query Log, and Query Dimension. In each situation, we re-evaluate the **linear score of description** (D-l for brief in Table 7), and the **NDCG@3 of description candidate list** (NDCG for brief in Table 7). The ablation results are shown in Table 7, where bold values indicate the most significant decline. It can be seen that, in all three situations, the results of ablation models underperform SRQG-Rule model with all features, confirming that every feature used in SRQG-Rule is necessary for extracting high-quality descriptions. Specifically, in "Overall Good" and "Query Log" situations, f_E is most important for finding top-1 description, and f_i is most important for extracting description candidate list. In fact, there are more queries and items in "Overall Good" and "Query Log" data that can be found in *"IsA"-like relation knowledge bases*, compared with "Query Dimension" data. Therefore, entity feature f_E plays a more important role in the linear score of the top-1 returned description. Besides, inclusion feature f_i is helpful to find *more specific descriptions* by improving their scores and bringing them to the front of returned description list, therefore it is vital for improving NDCG@3 of description candidate lists. In the "Query Dimension" situation, list feature f_L is most important for both the top-1 description and description candidate list. This is because items corresponding to a query in the "Query Dimension" situation are extracted from *list structures in HTML*, thus f_L has a greater impact on the results. Besides, distance feature f_d and frequency feature f_f also play an important role in all three situations for improving the quality of extracted descriptions.

5.8 Case Study

To intuitively compare different methods mentioned in this paper, we select four queries in MIMICS dataset and generate corresponding clarifying questions with different algorithms. Table 8 shows some example outputs of each algorithm, where RTC and QLM are two baselines, and S- indicates our proposed SRQG- algorithms for brief. For the query "google chrome exe", RTC and QLM cannot give a description of the query and items but all of our algorithms can extract a correct description for items from top results, like "*bit version of chrome*". For the second "quiet riot", both RTC and QLM generate "band" as a description of the query, but it is still not specific enough. By comparison, "metal band" or "heavy metal band" is more specific for the query "quiet riot" generated by our proposed algorithms. For the third example query "facial tingling", QLM generalizes RTC by generating a description "disease" for the query, and SRQG-LTR and SRQG-Gen generalize SRQG-Rule by selecting more specific descriptions for items. For the last *tail* query "cabins in asheville nc", RTC gives a generic clarifying question, but our proposed three algorithms all give wrong descriptions for query or items. This is because words or phrases like "owner" and "family home" are likely to appear frequently nearby the query or its corresponding items on many housing rental websites.

5.9 Discussion

In this paper, we use *template-based* clarifying questions as the basis. This is because existing conversational search scenarios lack manually written questions for training. In this circumstance, the benefit of template-based questions is that they have strong robustness and are easy to be applied to online systems. However, it has one obvious limitation: the template-based methods determine that we can only focus on the *slots* that need to be filled in one template (such as QUERY_DESC and ITEMS_DESC). However, a clarifying question can also be improved from the parts other than these slots. For example, for the query "GTA game" and the items "[Windows 7, Windows 8, Windows 10, Windows XP, Windows Vista]", existing methods will generate "Which operating system are you looking for?" as the question. However, if someone were asked to write this question manually, we will naturally think of **playing** the game, so the question may be written as "Which operating system do you want to *play GTA on*?". The enrichment of the other parts other than the slot also improves the readability and informativeness of clarifying questions, which is restricted by template-based approaches in this paper. One possible solution is to explicitly incorporate a user's potential **intent** into a question, such as "play the game" in the above example.

6 CONCLUSION

In this paper, we utilize top search results to generate more accurate and specific descriptions for a query and its corresponding items, thereby generating clarifying questions with higher quality for conversational search systems. We first design a rule-based unsupervised algorithm SRQG-Rule to select candidates for query and items from top result pages, then we design an ensemble ranking model SRQG-LTR and a neural model SRQG-Gen to generalize SRQG-Rule and further improve the quality of clarifying questions. We ask three trained annotators to evaluate descriptions and clarifying questions generated by our proposed methods. The experimental results prove that top search results are useful to find descriptions of queries and items because they contain abundant contextual information, therefore the quality of questions can be improved based on found descriptions. In the future, we aim to extract or generate items corresponding to a query from top search results and to generate a clarifying question simultaneously. In this way, clarifying questions can better help users filter top results, and finally find their intents in conversational search systems.

ACKNOWLEDGMENTS

Zhicheng Dou is the corresponding author. This work was supported by the National Natural Science Foundation of China No. 61872370 and No. 61832017, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. We also acknowledge the support provided and contribution made by the Public Policy and Decision-making Research Lab of Renmin University of China.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *International Conference on Learning Representations*.

[3] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 385–394.

[4] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*. Ieee, 1–6.

[5] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, et al. 2020. ConvAI3: Generating Clarifying Questions for Open-Domain Dialogue Systems (ClariQ). *arXiv preprint arXiv:2009.11352* (2020).

[6] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, et al. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.

[7] Pavel Braslavski, Denis Savenkov, Eugene Agichtein, et al. 2017. What do you mean exactly? Analyzing clarification questions in CQA. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*. 345–348.

[8] Lars Buitinck, Gilles Louppe, Mathieu Blondel, et al. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[10] Zhicheng Dou, Sha Hu, Yulong Luo, et al. 2011. Finding dimensions for queries. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 1311–1320.

[11] Zhicheng Dou, Zhengbao Jiang, Sha Hu, et al. 2015. Automatically mining facets for queries from their search results. *IEEE Transactions on knowledge and data engineering* 28, 2 (2015), 385–397.

[12] Zhicheng Dou, Zhengbao Jiang, Jinxiu Li, et al. 2017. A method of mining query facets based on term graph analysis. *Chines Journal of Computers* 40, 3 (2017), 556–569.

[13] Helia Hashemi, Hamed Zamani, and W Bruce Croft. 2020. Guided Transformer: Leveraging Multiple External Sources for Representation Learning in Conversational Search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1131–1140.

[14] Ayyoob Imani, Amir Vakili, Ali Montazer, et al. 2019. Deep neural networks for query expansion using word embeddings. In *European Conference on Information Retrieval*. Springer, 203–210.

[15] Bernard J Jansen, Danielle L Booth, and Amanda Spink. 2008. Determining the informational, navigational, and transactional intent of Web queries. *Information Processing & Management* 44, 3 (2008), 1251–1266.

[16] Zhengbao Jiang, Zhicheng Dou, and Ji-Rong Wen. 2016. Generating query facets using knowledge bases. *IEEE Transactions on Knowledge and Data Engineering* 29, 2 (2016), 315–329.

[17] Zhengbao Jiang, Zhicheng Dou, Wayne Xin Zhao, et al. 2018. Supervised search result diversification via subtopic attention. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1971–1984.

[18] Weize Kong and James Allan. 2013. Extracting query facets from search results. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 93–102.

[19] Weize Kong and James Allan. 2014. Extending faceted search to the general web. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 839–848.

[20] Antonios Minas Krasakis, Mohammad Aliannejadi, Nikos Voskarides, et al. 2020. Analysing the Effect of Clarifying Questions on Document Ranking in Conversational Search. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 129–132.

[21] Jiongnan Liu, Zhicheng Dou, Xiaojie Wang, et al. 2020. DVGAN: A Minimax Game for Search Result Diversification Combining Explicit and Implicit Features. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 479–488.

[22] Christopher D Manning, Mihai Surdeanu, John Bauer, et al. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[23] Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. 2020. Using BERT and BART for Query Suggestion. In *Joint Conference of the Information Retrieval Communities in Europe*, Vol. 2621. CEUR-WS. org.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[25] Xubo Qin, Zhicheng Dou, and Ji-Rong Wen. 2020. Diversifying Search Results using Self-Attention Network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1265–1274.

[26] Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*. 117–126.

[27] Sudha Rao and Hal Daumé III. 2018. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2737–2746.

[28] Sudha Rao and Hal Daumé III. 2019. Answer-based Adversarial Training for Generating Clarification Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 143–155.

[29] Julian Seitner, Christian Bizer, Kai Eckert, et al. 2016. A Large DataBase of Hypernymy Relations Extracted from the Web.. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 360–367.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017), 5998–6008.

[31] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[32] Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, et al. 2017. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2187–2193.

[33] Jian Wang and Wenjie Li. 2021. Template-guided Clarifying Question Generation for Web Search Clarification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3468–3472.

[34] Xiaojie Wang, Ji-Rong Wen, Zhicheng Dou, et al. 2017. Search result diversity evaluation based on intent hierarchies. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (2017), 156–169.

[35] Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, et al. 2015. An inference approach to basic level of categorization. In *Proceedings of the 24th acm international on conference on information and knowledge management*. 653–662.

[36] Wentao Wu, Hongsong Li, Haixun Wang, et al. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 481–492.

[37] Hamed Zamani, Susan Dumais, Nick Craswell, et al. 2020. Generating clarifying questions for information retrieval. In *Proceedings of The Web Conference 2020*. 418–428.

[38] Hamed Zamani, Gord Lueck, Everest Chen, et al. 2020. Mimics: A large-scale data collection for search clarification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3189–3196.

[39] Hamed Zamani, Bhaskar Mitra, Everest Chen, et al. 2020. Analyzing and Learning from User Interactions for Search Clarification. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1181–1190.