Information Retrieval Journal (2023) 26:9
https://doi.org/10.1007/s10791-023-09427-0

DeepQFM: a deep learning based query facets mining method

Zhirui Deng[1,2] · **Zhicheng Dou**[1,2] · **Ji-Rong Wen**[1,2]

Received: 10 October 2023 / Accepted: 12 October 2023 / Published online: 30 October 2023

Abstract
Search results from the search engine may be not enough to satisfy users' search intent when the issued query is broad or ambiguous. In such cases, presenting to the user *query facets*, which include common query reformulations, may help disambiguate the current query, save the effort of query reformulation, and improve the user's search experience. Existing approaches for mining query facets are mainly based on rule-based statistical features, but ignore the deep semantic information which can measure the relationship between items in facets more precisely and find more potential facet items. In this paper, we introduce a deep learning model with contrastive learning for query facets mining—DeepQFM. We first extract items from search result documents, form lists containing items having a parallel structure, and weight these lists based on their importance. Then, we cluster the weighted lists based on their semantic distance. Finally, we train an item encoder with contrastive sampling and rank the facets and the facet items based on their semantic representation. Experimental results show that our deep query facets mining model outperforms the state-of-the-art approach QDMiner in almost all evaluation metrics, especially for the recall and rp-nDCG, suggesting that DeepQFM can effectively mine more facet items from search result documents.

Keywords Query facets mining · Clustering · Contrastive learning · Deep learning

✉ Zhirui Deng
zrdeng@ruc.edu.cn

Zhicheng Dou
dou@ruc.edu.cn

Ji-Rong Wen
jrwen@ruc.edu.cn

1 Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, People's Republic of China

2 Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing 100872, People's Republic of China

1 Introduction

Search engines return web links related to an issued query. Usually, the user's query is short and ambiguous (Silverstein et al., 1999; Dou et al., 2007; Jansen et al., 2000; Song et al., 2007), leading to the user being dissatisfied with the search results (Shum, 2011; Niccolai, 2009). For example, when the user's query is *watches*, she may not only want to know information about *watch brands*, but also *watch materials* or *watch colors*. Besides, users' search queries are usually consistent in a search session. Users tend to search related topics after a search behavior. For example, a user prefers to search *man's watch*, *sports watch*, etc. after searching *watches*. Query facets mining aims to tackle these problems. A query facet is a category of a query's attributes. For example, for the second query *watches* in Table 1, it has three query facets which describe brands, color and material and one of its query facets is {*cartier, baume mercier, tag heuer, rolex*} which describes brands. There are several facet items in each query facet. For the above query *watches*, *cartier*, *baume mercier*, etc. are called the facet item. When a user's query is too broad, such as *watches*, she will often reformulate it to a more specific query, such as *cartier watches*. Query facets provided in search engine shortcut the reformulation process, by inferring what a user might be interested in when the query is broad and ambiguous, saving the cognitive load of coming up with the reformulation.

Existing supervised and unsupervised query facets mining approaches (Dou et al., 2011, 2015; Jiang et al., 2016) are primarily based on statistical features. Dou et al. (2011) first proposed query dimensions that have the same meaning as query facets. The model proposed by Dou et al. solely used statistical features such as IDF, rank, word intersect count, etc. to obtain the weight of the extracted lists and then modified cluster algorithm QT as WQT to get the final query facets. Kong proposed two supervised models: QF-I and QF-J (Kong & Allan, 2013). Both models used a Bayesian network to judge whether an item was a facet item and whether two facet items were in one query facet. The item representation constructed in QF-I and QF-J was also based on statistical features. Jiang et al. (2016) used items extracted from the knowledge base additionally and followed QDMiner's routine to cluster the items and rank query facets.

Previous studies about query facets mining have proved that statistical features, such as TF, IDF, rank, etc. could be used to improve the performance of query facets mining models. However, these statistical features based methods ignore the importance of deep

Table 1 Examples of query facets

Query	Query facets
Mobile phones	Samsung galaxy s, Iphone 4, Nokia n97,...
	Nokia, Blackberry, Motorola, Apple,...
	Memory card reader, mp3, alarms,...
Watches	Cartier, baume mercier, tag heuer, rolex,...
	Black and white, pink, army green, silver,...
	Precious metal, brass, amber, ruby,...
Chinese restaurants	Sichuanese cuisine, peking restaurant,...
In seattle redmond	Capitol hill, ballard, university district,...
	American, Asian, Japanese, Italian,...
US universities	Harvard University, Princeton University,...
	Business, design, law, engineering,...
	Sat, toefl, gmat, gre, ielts,...

semantic information among items. Factually, the deep semantic information among items is prominent for query facets mining. For example, *Rolex air king* rarely appears in search results, thus its statistical features may be the same as unimportant items such as *Wisconsin* and solely using statistical features is hard to select it. Combined with deep semantic information, it is close to the query *watches* in semantic space and can be extracted as a facet item. Recently, pre-trained language models have proved effective for various downstream tasks. Inspired by these, in this paper, we leverage both deep semantic features and statistical features to better measure the semantic information of an item and the relationship between items in order to help recall more facet items from a candidate item set. Moreover, we will focus on utilizing contrastive learning to optimize deep semantic features and mine query facets from search result documents without introducing other knowledge bases.

We propose a Deep Query Facets Miner (DeepQFM) to mine query facets. Our model is a pipeline framework with contrastive learning. Except for statistical features, we leverage deep semantic features to further understand the item's semantic information. We adopt contrastive learning to enhance data representation, so that our model can better measure the relationship between items and recall more facet items from the candidate item set. More specifically, our model consists of four steps. *First*, because items in a query facet have a parallel structure, we extract lists containing items with a parallel structure from search result documents. Then, we weigh these extracted lists based on their relevance to the query. *Second*, we design an item-contrastive sampling that constructs item pairs to generate self-supervised signals. The contrastive learning task is designed to pretrain the item encoder and learn better item representation. Moreover, we also devise a list-contrastive sampling to provide an ablation study with our list encoder based on BERT. *Third*, a single list does not contain all facet items, so we design WQT_{Deep} algorithm to cluster similar lists based on the BERT representation of these lists. Finally, we rank facets and facet items based on their importance score, which consists of statistical features and deep semantic features, and discard unimportant items to obtain the final query facets.

Experimental results show that our model outperforms existing approaches in 10 out of 12 metrics, especially the recall of facet items. This demonstrate that our deep learning method can better measure the semantic meaning of an item and the relevance between items.

In summary, the major contributions are three-fold:

- We propose a deep learning pipeline framework with contrastive learning to mine query facets. As far as we know, this is the first time that deep semantic features have been applied to query facets mining.
- We design a self-supervised learning task to learn a better representation of items. Based on the representation, we can measure the relationship between items in deep semantic space more accurately, so that we can recall more facet items from search result documents. Besides, we also propose a list-contrastive sampling approach to perform an ablation study with our list encoder based on BERT.
- We combine deep semantic features with rule-based statistical features to provide a better semantic matching score so that similar items can be grouped together.

The rest of this paper is organized as follows. Section 2 discusses the related work. In Sect. 3, we introduce our model DeepQFM. Section 4 describes the dataset and the evaluation metrics we used. Experimental results are shown in Sect. 5. Section 6 is the conclusion.

2 Related work

There are many fields related to query facets mining. In this section, we will briefly introduce works in these fields and summarize the relationship between DeepQFM and them.

2.1 Query facets mining and faceted search

Query facets mining is to mine groups of facet items from search result documents. Each group of facet items describes an aspect of the query. These facet items can provide users with more information and they can also be used in search result diversification (Hu et al., 2015) and clarifying question generation (Aliannejadi et al., 2019). Dou et al. (2011), Dou et al. (2015) first proposed QDMiner to mine query facets from search result documents. QDMiner was an unsupervised method and it extracted item lists from search result documents and used statistical features contained in lists such as IDF and document ranking to weight lists. After this, they proposed WQT (Quality Threshold with Weighted data points) for clustering and finally ranked facets and facet items. QF-I/QF-J (Kong & Allan, 2013) was a supervised method. Kong et al. used a Bayesian network to classify whether an item was a facet item and whether two items should be grouped together. With the help of the above two labels, Kong et al. clustered and ranked the items to get the final query facets. Both above two methods only used statistical features such as TF, IDF, rank, etc. to model the importance of the item and the relationship between items while the representation and the similarity of items in semantic space play an important role in query facets mining. Therefore, some deep semantic features should be leveraged to improve the performance of the model.

Query facets mining is to generating facets in open-domain while some traditional faceted search methods are domain specific, such as product search (Dakka et al., 2005, 2006; Stoica et al., 2007; Basu Roy et al., 2008; Dakka & Ipeirotis, 2008; Dash et al., 2008; Ben-Yitzhak et al., 2008; Diao et al., 2010; Li et al., 2010; Latha et al., 2010; Pound et al., 2011). For example, Dakka and Ipeirotis (2008) automatically mined facets with an unsupervised technique that was used to browse text databases. Facetedpedia (Li et al., 2010) extracted and generated facets from Wikipedia which was a specific knowledge database.

2.2 Query recommendation

Query recommendation is to suggest queries for users when sometimes the query is ambiguous or broad. It can help users find information more precisely. Existing methods for query recommendation were to generate queries that were similar to the original query in semantic meanings (Mitra et al., 1998; Anick, 2003; Baeza-Yates et al., 2004; Zhang & Nasraoui, 2006; Riezler et al., 2008; Huang & Efthimiadis, 2009; Herdagdelen et al., 2010; Szpektor et al., 2011; Xue & Croft, 2013; Li et al., 2013; Bing et al., 2015). Users could select one of the generated queries to get better search results. Different from query recommendation, query facets mining aims to mine facet items that describe different aspects of the query instead of finding information related to the query. Besides, facets mined by the query facets mining model are structural and describe different topics of the query. Sometimes, the results can be used for query recommendation.

2.3 Query-based summarization

Summarization of text is a basic task in natural language processing. Existing works about text summarization can be classified into different types according to summary construction methods (abstractive/extractive), number of document sources (single/multiple documents), and information types in the summary (indicative/informative) (Gholamrezazadeh et al., 2009; Damova & Koychev, 2010; Maxwell et al., 2017). Recently, researchers proposed to leverage query aspects to summarize search results (Sarwar et al., 2021; Surya et al., 2021). Although query facets are also a summarization of search result documents, there are two differences between them. First, text summarization uses sentences from search result documents while query facets mining uses lists extracted from search result documents. Besides, the result of text summarization is a flat list of sentences, while the results of query facets mining are sets of semantically related items.

2.4 Query subtopics mining

Query subtopics mining identifies subtopics of queries in search engine interaction logs (Jones & Klinkner, 2008; Strohmaier et al., 2009; Hu et al., 2012) or search result documents (Radlinski et al., 2010; Liu et al., 2015). Different from query facets, subtopics usually do not have a hierarchical structure, and solely consist of several items, where similar items are not grouped together. For example, the query *apple* has subtopics *apple fruit* and *apple company*. Query facets are similar to query subtopics, but they are hierarchical. For example, the query *apple* has facets such as [*green apple*, *apple tree*] and [*ipad*, *iphone*] associated with subtopics of the query. Both query facets mining and query subtopics mining can be applied to search result diversification (Sarwar et al., 2020; Hu et al., 2015). However, query facets have a hierarchical structure and can better diversify search results than query subtopics (Hu et al., 2015).

2.5 Semantic representation

Representing a word as a semantic embedding has been adopted by many deep learning models (Mikolov et al., 2013a, b). Therefore, it is particularly important to get a good word embedding which needs to contain as much useful information as possible, for example, the contextual information of the word, and the knowledge related to the word (Devlin et al., 2018; Li et al., 2015; Melamud et al., 2016; Sun et al., 2020; Zhao et al., 2017). Context2vec used LSTM to force word embedding containing the contextual information of the word (Melamud et al., 2016). Ngram2vec introduced ngram into four representation methods and learned improved word representations that could better reflect their semantic meanings (Zhao et al., 2017). BERT utilized the transformer as the main framework to train a pretrained model on a large corpus (Devlin et al., 2018) and the model achieved state-of-the-art results on eleven NLP tasks. CoLAKE added knowledge to the word representation based on BERT (Sun et al., 2020).

In query facets mining, we combine the semantic representation with the statistical features to get a better representation of the item. Then, we use this representation for subsequent clustering and ranking.

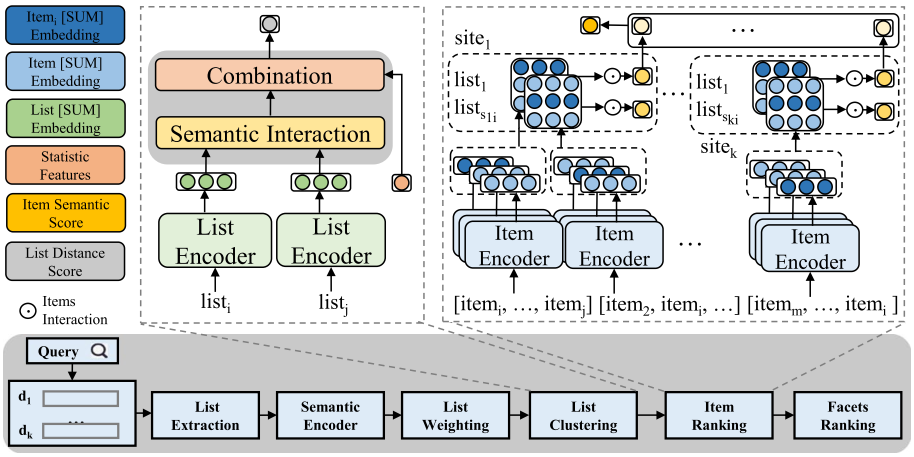

Fig. 1 Model architecture

3 Model

In this section, we will restate the problem, present the structure of our model DeepQFM, and show how we use DeepQFM to mine query facets.

Problem Definition Query facets mining is to mine query facets $\mathcal{F}$ from top k search result documents for a given query q. A query facet $f \in \mathcal{F}$ is a set of facet items $\{i_1, i_2, \dots, i_n\}$ which describe an aspect of the query. For example: for query *mobile phones* shown in Table 1, there are three query facets, one of them is {*nokia*, *blackberry*, *motorola*, *apple*} and *blackberry*, *samsung galaxy s*, etc. are facet items.

Our model follows four steps: list extraction and weighting, deep semantic encoders, list clustering, and facet and item ranking. The overall architecture of DeepQFM is shown in Fig. 1. In this paper, we mainly focus on deep semantic encoders, list clustering, and item ranking which integrate deep semantic features to mine query facets.

- *List Extraction* Items in a query facet have a parallel structure. Therefore, we first extract items with a parallel structure from the search result documents to form lists. For example, *.albert einstein, anne frank, aristotle,...* will be extracted as lists for the query *quotes and sayings*. Besides, we proposed repeat context patterns to extract more item lists from search result documents.
- *List Weighting* We weight lists according to their relevance to the query. Following Dou et al. (2011), we leverage statistical features to assign low weight to noisy lists such as [*to order email us, monday friday 9, or call us*] for query *watches*.
- *Deep Semantic Encoders* We encode items and lists with the pretrained model to generate better data representation. To leverage deep semantic information and provide better representation for semantic matching, we design a contrastive sampling method to train a better item encoder.
- *List Clustering* A single list is not enough to reflect the whole content about an aspect of a query. Therefore, lists which have similar semantic meanings are grouped together, according to their semantic representation distance in semantic space.

- *Facet Ranking and Item Ranking* It is necessary to rank the clusters above based on their importance to obtain the final query facets. Thus, following Dou et al. (2011), we rank the cluster according to the weight of the lists in the cluster. Besides, there are still some noisy items in the query facets. With deep semantic features, we can rank the items in a facet to remove these noisy items and obtain the final query facets.

3.1 List extraction

As stated above, a query facet consists of items with parallel structures. However, we solely have the top retrieved documents from the search engine. Following Dou et al. (2011), we extract lists of items from the search results based on three types of patterns: text patterns, HTML patterns and repeat region patterns. We add repeat context patterns to extract more useful lists from the search results.

Text patterns extract text with two regular expression forms: item{, item} (and| or) {other} item and {^item(:| -).+$}$^+$. Examples of the two patterns are shown below, wherein the extracted lists is italicized. Example 1 extracts items connected by **,**|and|or and Example 2 extracts parallel items followed by :| -.

Example 1 NOW—Science headline news from all realms of science, including *biology, genetics, medicine, stem cells, evolution, animals, climate change, the environment, physics, astronomy, and science policy.*

Example 2 *Computer* loads the drives list...*Documents*: loads user's documents...*Network*: loads a list of all network clients connected...

HTML patterns extract text in four types of HTML tags: <select>, <ul>, <ol> and <table>. Items in these four parallel structures are extracted as lists. Examples of the four HTML patterns are shown below. Items below will be extracted as a list. As for the Type 4, each row and each column are extracted as a list, respectively. That is, if a table has m rows and n columns, then we extract at most m+n lists.

Type 1 <select><option>item$_1$ </option>
<option>item$_2$ </option>...</select>
Type 2 <ul><li>item$_1$ </li><li>item$_2$ </li>...</ul>
Type 3 <ol><li>item$_1$ </li><li>item$_2$ </li>...</ol>
Type 4 <table><tr><td>item<td>...</table>

Repeat region patterns contain at least two adjacent or nonadjacent repeat regions in webpages based on DOM trees. All leaf HTML nodes in repeat regions are extracted and grouped by tag names and display styles. An example is shown in Fig. 2. It can be found from the example that the flower's name, the price, and the type of flower can be extracted as three lists because items in them have a parallel relationship. Thus, we can extract three lists: the name of the flower (*Harvest Sunflower Basket*, *Hello Sunshine Bouquet*, *Light of My Life Bouquet*), the price (*$85-$125*, *$56-$92*, *$50-$80*), and the type of flower (*Local Florist Crafted*, *Local Florist Crafted*, *Local Florist Crafted*). (Images are out of scope of this work.)

In addition to the above three patterns, we extract items in the same list which have the same prefix or suffix, and name them repeat context patterns to mine more lists from search

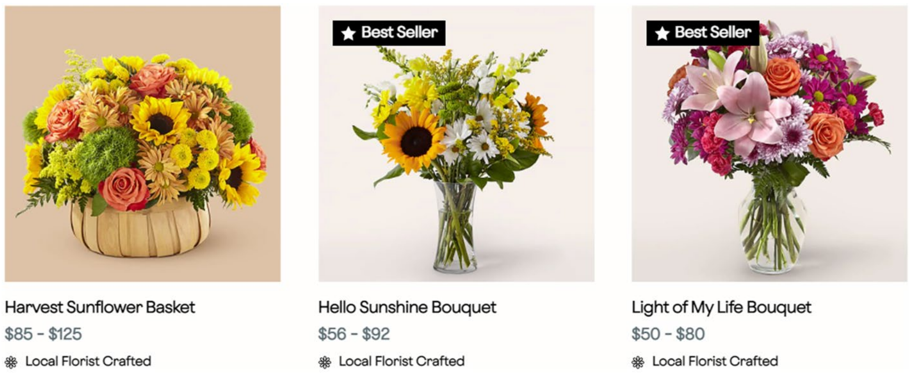

Fig. 2 An example of repeat region pattern

result documents. For example, in list *tag heuer watches, movado watches, rotary watches, wyler watches, glam rock watches, seiko watches*, all items have the same suffix *watches*. So we extract a new list *tag heuer, movado, rotary, wyler, glam rock, seiko*. In addition, lists that have a structure of numbers followed by English or English followed by numbers will also be extracted. For example, in the list *accessories watches 2, activa watches 75, baume mercier 13, bulova watches 24, caravelle by bulov 8, cartier watches 12, diesel watches 12*, we can extract a new list: *accessories watches, activa watches, baume mercier, bulova watches, caravelle by bulov, cartier watches, diesel watches*. Types of repeat context patterns are summarized in Table 2.

We also extract the list's context which is the tokens before the list. After completing the above steps, following Dou et al. (2011), we further process by removing useless symbol characters such as '[' and ']' and converting uppercase to lowercase. Second, items longer than 20 words are removed. Finally, lists containing fewer than 2 items or more than 200 items are removed.

3.2 List weighting

There are some noisy lists in the extracted list set, as shown in Table 3. For the query *watches*, the first example is non-relevant and the second example is an extraction error where the number *13* and *8* should not be extracted. In this section, we assign weight to all lists extracted in Sect. 3.1 according to their importance and assign lower weights to lists that are not relevant to the query or extraction errors.

In order to measure the importance of lists and remove bad lists, following Dou et al. (2011), we use the product of document matching weight S_{doc} and average inverse document frequency S_{idf} to calculate the score of a list as $S_l = S_{doc} \times S_{idf}$. If a list contains more items that appear in the higher-ranked documents, it is more important and should be given a higher score. Thus, the definition of S_{doc} is $\sum_{d \in D} \left(\frac{|l \cap d|}{|l|} \times \frac{1}{\sqrt{\text{rank}_d}} \right)$, where D is the search result document set, $|\, l \cap d \,|$ is the number of items that appear in both list l and document d, $|\, l \,|$ is the number of items in list l, rank_d is the rank of document d. Besides, lists containing more common items in a corpus are also less informative and should be assigned low scores. Based on the above consideration, S_{idf} is defined as $\frac{1}{|l|} \times \sum_{i \in l} log \frac{N - N_i + 0.5}{N_i + 0.5}$, where

Table 2 Types of repeat context patterns

Pattern type	Extracted list
Prefix item_1, prefix item_2,...	Item_1, item_2,...
Item_1 suffix, item_2 suffix,...	Item_1, item_2,...
Number_1 item_1, number_2 item_2,...	Item_1, item_2,...
Item_1 number_1, item_2 number_2,...	Item_1, item_2,...

Table 3 Example of noisy lists (query: watches)

Type	Noisy lists
Non-relevant	60 Items per page,..., 120 items per page
Error	Baume mercier 13,..., caravelle by bulov 8

N is the total number of documents in the corpus, N_i is the total number of documents containing item_i. We use ClueWeb09 to calculate N and N_i.

3.3 Deep semantic encoders

To further improve the quality of query facets, it is important to learn better data representation. However, there are only 190 labeled training examples for query facets mining and thus data sparsity is a major challenge for most query facets mining models, especially for neural models. Motivated by this, we first design a contrastive sampling approach for items and leverage the sampled item pairs to pretrain the item encoder. In addition, for lists, we leverage the general language model BERT to generate list embeddings.

3.3.1 Contrastive sampling

To provide a better representation for semantic matching and improve the performance of our model, we design an item-contrastive sampling strategy to generate item pairs. To verify the effectiveness of self-supervised learning in the lists' representations, we also design a list-contrastive sampling approach to perform an ablation study.

Item Pair Items in the same facets tend to have similar semantics while items in different facets are more likely to cover different aspects of the query. Therefore, we construct item-supervised signals to close the representation of items in the same facet and pull away items in different facets.

There are several facets in the label of query q. For example, for query *watches* in Table 1, there are three facets. We regard two items in the same query facet of the label data such as *cartier* and *rolex* as the positive pair and items in the same mini-batch as the negative samples I^-. We use BERT as the initial item encoder because it is pretrained on a large corpus and other advanced pretrained models can also be used. We fine-tune the item encoder with item pairs and obtain an item encoder tailored for query facets mining to learn better item representation, shown in Eqs. (1, 2).

$$r_{i_1} = \text{BERT}_{[\text{SUM}]}(i_1), \quad i_1 = [\text{CLS}]\text{item}_1[\text{SEP}], \tag{1}$$

$$r_{i_2} = \mathrm{BERT}_{[\mathrm{SUM}]}(i_2), \quad i_2 = [\mathrm{CLS}]\mathrm{item}_2[\mathrm{SEP}]. \tag{2}$$

Formally, the loss function of item augmentation $\mathcal{L}_{IP}$ can be defined as:

$$\mathcal{L}_{IP} = -\log \frac{\exp(\mathrm{Sim}(r_{i_1}, r_{i_2}))}{\exp(\mathrm{Sim}(r_{i_1}, r_{i_2})) + \sum_{i^- \in I^-} \exp(\mathrm{Sim}(r_{i_1}, r_{i^-}))}, \tag{3}$$

where the function $\mathrm{Sim}(\cdot)$ is the cosine similarity which can be replaced by other similarity measure methods such as the inner product.

List Pair In addition to item-contrastive sampling, it is important to learn a list representation that can reflect the meaning of items in the list and better measure the similarity and differences between different lists. To implement this idea, we design list-contrastive sampling methods to construct augmented list pairs based on lists extracted in Sect. 3.1. The representations of augmented list pairs from one list should be closer than augmented lists from different lists.

Specifically, we design three list augmentation strategies. (1) Item deleting: this strategy randomly deletes some items from the list to enhance the generalizability of the encoder. (2) Item reorder: this method randomly swaps two items in a list, as changing the order of items in the list does not usually affect the overall meaning of the list. (3) Representation dropout: Following Gao et al. (2021), we pass a list through the same model twice with two different dropouts and get two different list representations.

For list l, we apply two random augmentation strategies on the list and get two lists l_1 and l_2. We concatenate items in the list and use the list encoder to learn the representation of lists r_{l_1} and r_{l_2}, shown in Eqs. (4, 5). Similar to the item encoder, we use BERT as the initial list encoder and fine-tune it based on the above list pairs.

$$r_{l_1} = \mathrm{BERT}_{[\mathrm{SUM}]}(l_1), \quad l_1 = [\mathrm{CLS}]\mathrm{item}_1[\mathrm{SEP}] \ldots [\mathrm{SEP}]\mathrm{item}_n[\mathrm{SEP}], \tag{4}$$

$$r_{l_2} = \mathrm{BERT}_{[\mathrm{SUM}]}(l_2), \quad l_2 = [\mathrm{CLS}]\mathrm{item}_1[\mathrm{SEP}] \ldots [\mathrm{SEP}]\mathrm{item}_n[\mathrm{SEP}]. \tag{5}$$

r_{l_1} and r_{l_2} is regarded as the positive pair. The augmented lists from other lists in the same mini-batch are regarded as the negative samples L^-. Similarly, the loss function of list augmentation pair $\mathcal{L}_{LP}$ for two lists is:

$$\mathcal{L}_{LP} = -\log \frac{\exp(\mathrm{Sim}(r_{l_1}, r_{l_2}))}{\exp(\mathrm{Sim}(r_{l_1}, r_{l_2})) + \sum_{l^- \in L^-} \exp(\mathrm{Sim}(r_{l_1}, r_{l^-}))}. \tag{6}$$

3.3.2 List and item encoding

Through the above contrastive learning, we have pretrained an item encoder which is suitable for query facets mining domain. For the list encoder, we use the pretrained model BERT without self-supervised learning as our main approach. The above list encoder with contrastive learning is used to perform an ablation study on whether the self-supervised learning on lists is useful. We embed the lists and items with the item encoder and the list encoder, respectively and use the vectors as their deep semantic features. We will combine these deep semantic features with statistical features in the following steps to recall more facet items from the candidate item set.

3.4 List clustering

A single list cannot be used as a query facet because (1) Lists extracted in Sect. 3.1 are not complete. Most lists only contain part of the information in a query facet and cannot be used as a query facet directly. (2) There are some noisy lists and we need to remove these lists. (3) Some lists contain duplicated items. Due to the above reasons, we need to cluster similar lists and discard noisy lists. To provide a better representation for semantic matching and recall more facet items, in this section, we integrate deep semantic information into the list clustering process.

Compared with other clustering algorithms, WQT (Dou et al., 2011) inherits the strength of QT algorithm which guarantees the quality of clusters and does not need to specify the number of clusters which is suitable for query facets mining. Besides, WQT modifies QT algorithm which regards all data as equally important, and suits our scenario where lists are not equally important. Previous methods (Dou et al., 2011; Kong & Allan, 2013) only use statistical features, such as the number of overlap items in two lists, to calculate the distance between two clusters. However, some lists which have similar semantic meanings but no overlapping items cannot be grouped together. For example, the list: *acer, alcatel, apple, archos, asus* and the list: *nokia, htc, samsung, google* have no identical items but the items in the two lists are all brands and have same semantic meanings. To group lists with the same semantic meanings, we modify the WQT algorithm and proposed a deep version WQT_{Deep}, shown in Algorithm 1.

First, we choose the list l which has the highest weight from the weighted list set T_L obtained in Sect. 3.2, and set it as the initial cluster c_i. Then, we need to find the list l_{min} which is the closest to cluster c_i and their minimum distance $d_{l_{min}}$ is defined in Eq. (7). The distance between a list and a cluster $\mathrm{dis}_{l_k c_i}$ is defined in Eq. (8), that is the maximum distance between list l_k and each list l_j in c_i.

$$d_{l_{min}} = \min_{l_k \in T_L}(\mathrm{dis}_{l_k c_i}), \tag{7}$$

$$\mathrm{dis}_{l_k c_i} = \max_{l_j \in c_i}(\mathrm{dis}_{l_k l_j}). \tag{8}$$

Because we want to combine statistical features with deep semantic features for more accurate semantic matching, the distance between lists $\mathrm{dis}_{l_k l_j}$ can be defined as a linear combination of the semantic distance and the statistical distance, shown in Eq. (9). The semantic distance $\mathrm{semdis}_{l_k l_j}$, shown in Eq. (10), measures the distance between two lists' representation in the semantic space, so it is calculated by the cosine distance of list_k's representation R_{L_k} and list_j's representation R_{L_j}. We use cosine distance because it is useful and easy to switch to other distance functions, such as Euclidean distance and Manhattan distance. The statistical distance, following Dou et al. (2011), shown in Eq. (11), is calculated by the items' intersection of two lists.

$$\mathrm{dis}_{l_k l_j} = (1-\lambda) \times \mathrm{semdis}_{l_k l_j} + \lambda \times \mathrm{stadis}_{l_k l_j}, \tag{9}$$

$$\mathrm{semdis}_{l_k l_j} = \left(1 - \frac{R_{L_k} \cdot R_{L_j}}{|R_{L_k}||R_{L_j}|}\right)/2, \tag{10}$$

$$\text{stadis}_{l_k l_j} = 1 - \frac{| l_k \cap l_j |}{\min(| l_k |, | l_j |)}. \tag{11}$$

If $d_{l_{min}} \leq D_{max}$ where D_{max} is a hyperparameter indicating the maximum diameter, we add the list l_{min} to cluster c_i. As each list is extracted from a search result document and has its own website and we set w_i=|site(c$_i$)| as the count of different site in cluster c_i. If $d_{l_{min}} > D_{max}$ and $w_i \geq W_{min}$, we save cluster c_i, remove all lists in cluster c_i from T_L and continue to find cluster c_{i+1}.

Algorithm 1 Deep Clustering Algorithm: WQT$_{\text{Deep}}$

```
Input: $T_L$: Weighted list set, $D_{max}$: Maximum diameter, a hyperparameter,
    $W_{min}$: Minimum site weight, a hyperparameter.
Output: $C = \{c_1, c_2, ..., c_n\}, c_i = \{\text{item}_1, ..., \text{item}_k\}$: Cluster result.
1: while $T_L \neq \emptyset$ do $d = 0$
2:     $l = \arg\max_l S_l := \{l \mid \forall l_i \in T_L, S_{l_i} \leq S_l\}$
3:     $c_i = \{l\}$
4:     while $T_L \neq \emptyset$ and $d <= D_{max}$ do
5:         $d_{l_{min}} = \min_{l_k \in T_L}(\text{dis}_{l_k c_i}) = \min_{l_k \in T_L}(\max_{l_j \in c_i}(\text{dis}_{l_k l_j}))$
6:         if $d_{l_{min}} \leq D_{max}$ then
7:             $c_i = c_i \cup \{l_{min}\}$
8:             $d = \max(d, d_{l_{min}})$
9:         else
10:            break
11:        end if
12:    end while
13:    $w_i$=| site(c$_i$)|
14:    if $w_i \geq W_{min}$ then
15:        Save the cluster $c_i$.
16:    end if
17:    $T_L = T_L$ - $c_i$
18:    Go to step 1
19: end while
20: return $C$;
```

3.5 Facet and item ranking

The result of list clustering in Sect. 3.4 is the cluster set C that has not been sorted among clusters. To help users find the information they need more quickly, in this section, we rank clusters in the cluster set C based on their importance to generate the final query facets.

Following Dou et al. (2011), we define the importance of each cluster S_c as Eq. (12). According to Algorithm 1, each cluster c is comprised of several lists and each list belongs to a website. There are some duplicated lists in each cluster, and most duplicated lists are on the same website. Therefore, for lists from the same website, we only calculate the list with the highest score. For each cluster, we sum the highest score of lists on different websites as the cluster score S_c and rank the cluster based on the score.

$$S_c = \sum_{s \in site(c)} \max_{l \in c, l \in s} S_l, \tag{12}$$

where $site(c)$ is the set of websites of lists in cluster c and s is a website. l is the list in both cluster c and site s and S_l is its score according to Sect. 3.2.

Similarly, items in a query facet should also be sorted, because: (1) there are still some noisy items in the facet and it is crucial to remove them; (2) ranking the items can satisfy users' information needs more quickly. Therefore, in this section, we remove the noisy item which is non-relevant to other items in a facet based on their deep semantic meanings, and rank the items in the same facet based on their importance.

An item can appear in several lists and items appearing in more lists tend to be more important. Besides, an item that is different from other items and has low scores in most lists tends to be a noisy item. Based on these considerations, we score item i in cluster c as Eq. (13). Because lists from the same website are more likely to be duplicated, we first average the item's score in lists from the same website and then sum the average item's score in different websites:

$$S_{i|c} = \sum_{s \in site(c)} \frac{1}{|L|} \sum_{i \in l, l \in c, l \in s} S_{i|l}, \tag{13}$$

where $site(c)$ is the website set of lists in cluster c, s is a website in $site(c)$, $|L|$ is the number of lists whose website is s and $S_{i|l}$ is the item i's score in list l.

Items in a list tend to have a parallel structure and similar semantics. An item that is not similar to other items in the same list tends to be a noisy item and should be assigned a lower weight. Therefore, we calculate the average of semantic similarity score of item i with other items in list l – sim_{ij} to score item i in list l, denoted as Eq. (14):

$$S_{i|l} = \frac{1}{|l| - 1} \sum_{j=1, j \neq i}^{|l|} \text{sim}_{ij}, \tag{14}$$

where $|l|$ is the number of items in list l.

As for the semantic similarity between two items sim_{ij}, we use the cosine similarity of two items' semantic embedding according to Eq. (15):

$$\text{sim}_{ij} = \frac{R_i \cdot R_j}{|R_i||R_j|}, \tag{15}$$

where R_i and R_j is the embedding of item_i and item_j learned in Sect. 3.3, respectively.

After sorting items according to their scores, to remove noisy items, following Dou et al. (2011), we only keep the items whose score $S_{i|c} > 1.0$ and $S_{i|c} > \frac{|\text{site}(c)|}{10}$, where $|\text{site}(c)|$ is the number of different websites of lists in cluster c. With above methods, we can get the final ranked query facets $\mathcal{F}$.

Table 4 The statistics of UserQ and RandQ

Raw data	UserQ	RandQ
Queries	83	105
Items	490490	476875
Results per query	99.8	99.3
Lists per document	17.6	14.3
Items per list	8.0	8.5
Label data	**UserQ**	**RandQ**
Good and fair facets per query	10.3	5.1
Items per facets	38.6	35.9
Items per query	399.9	181.7

4 Evaluation

4.1 Data

Following previous work (Dou et al., 2011), we use two datasets UserQ[1] and RandQ.[2] UserQ is composed of 83 queries issued by the subjects and RandQ is composed of 105 queries randomly sampled from a commercial search engine's query log. The top 100 result documents are extracted under each query. Each query has a corresponding manual label marked by persons who search in the query-related field or have relevant knowledge. Each label is composed of multiple manually classified query facets. Each facet has a facet name and a rate. The rate is divided into three types: good/2, fair/1, and bad/0.

The statistics of UserQ and RandQ are shown in Table 4. Both UserQ and RandQ contain more than 0.4 million items, and the average number of result documents contained in each query is 99.8 and 99.3. The average number of lists contained in each result document is 17.6 and 14.3, and the average number of items contained in each list is about 8.0 and 8.5. In addition, in the label data, the average number of good and fair facets contained in each query is 10.3 and 5.1. The average number of facet items contained in each facet is about 38.6 and 35.9, and the average number of facet items contained in each query is 399.9 and 181.7.

4.2 Evaluation metrics

The results of the query facets mining problem can be evaluated from four aspects: accuracy of finding items, quality of clustering, accuracy-clustering quality, and ranking effectiveness of query facets.

4.2.1 Accuracy of finding items

An important aspect of evaluating the results of query facets mining models is whether the model can find facet items that need to be retained from the extracted lists. Therefore,

[1] http://playbigdata.ruc.edu.cn/dou/qdminer/UserQ/.

[2] http://playbigdata.ruc.edu.cn/dou/qdminer/RandQ/.

according to Eqs. (16) and (17), we use precision and recall to measure the accuracy of the model for finding facet items.

$$\text{Precision}: \frac{\mid \{I\} \cap \{I^{lab}\} \mid}{\mid \{I\} \mid}, \tag{16}$$

$$\text{Recall}: \frac{\mid \{I\} \cap \{I^{lab}\} \mid}{\mid \{I^{lab}\} \mid}, \tag{17}$$

where $\{I\}$ represents the set of items extracted by the model. $\{I^{lab}\}$ represents the item set contained in the label data, that is, the ground truth.

4.2.2 Quality of clustering

Each query facet should only contain facet items reflecting the same aspect of the query. Facet items that contain similar information should not be divided into different facets. Therefore, similar to Dou et al. (2011), we use Purity, NMI (Normalized Mutual Information), RI (Random Index), and F measure as evaluation metrics to measure the quality of clustering.

4.2.3 Accuracy-clustering quality

To further evaluate the overall performance of accuracy and clustering of the model, we adopt PRF and wPRF proposed by Kong and Allan (2013). PRF is the multiplication of precision, recall, and cluster F1 score and wPRF is the multiplication of weighted precision, weighted recall, and weighted F1 calculated by weighted precision and weighted recall.

4.2.4 Ranking effectiveness of facets

In the final result, we want to rank good facets (and good facet items) before bad ones. Dou et al. (2011) proposed a modified version of nDCG: fp-nDCG and rp-nDCG. Based on the original formula of nDCG, both fp-nDCG and rp-nDCG have added a weight w_i. The calculation formulas of the three measurements are shown in Eq. (18).

$$\text{nDCG} = \frac{\text{DCG}}{\text{IDCG}}, \qquad \text{DCG} = \sum_{i=1}^{f} \text{DG}_i, \qquad \text{DG}_i = \frac{2^{r_i-1}}{log_2(1+i)}, \tag{18}$$

where the rating of facet f_i is r_i.

For fp-nDCG and rp-nDCG, we set:

$$\text{DCG} = \sum_{i=1}^{f} w_i * \text{DG}_i, \tag{19}$$

and w_i is calculated as follows:

$$\text{fp-nDCG}: w_i = \frac{|f_i \cap f_i^{label}|}{|f_i|}, \tag{20}$$

$$\text{rp-nDCG} : w_i = \frac{| f_i \cap f_i^{label} |}{f_i} \frac{| f_i \cap f_i^{label} |}{| f_i^{label} |}, \tag{21}$$

where f_i represents facet generated by the model, and f_i^{label} is the facet in label data.

4.3 Baselines

There are several approaches in query facets mining, and we compare our method with QDMiner (Dou et al., 2011) and QF-I/QF-J (Kong & Allan, 2013).

QDMiner followed four steps. It extracted lists from search result documents. Then the extracted lists were weighted according to statistical features. The lists were clustered based on the WQT clustering algorithm (described in Sect. 3.4). Finally, the model ranked the facets and the facet items based on statistical features.

QF-I/QF-J was a supervised method, starting with lists extracted as with QDMiner. It then classified whether an item was a facet item and whether two items should be grouped together based on a Bayesian network. Finally, it clustered and ranked the facet items and facets.

Note that a related algorithm (QDM_{KB} (Jiang et al., 2016)) employs a knowledge base to aid in facets mining, but as we focus on search results as the source of query facets, we do not compare to this work.

4.4 Implementation details

For the baselines, and for the proposed DeepQFM method, we extract lists from the top 100 results of UserQ and RandQ (described in Sect. 4.1). We use five-fold cross-validation to get more stable results. We tune parameters based on the sum of precision, NMI, nDCG, fp-nDCG and rp-nDCG, because precision reflects the proportion of real facet items among recalled facet items, NMI shows the clustering quality, and nDCG, fp-nDCG, and rp-nDCG measure the ranking quality. We pre-train the item encoder with two epochs and the learning rate is 5e–5. We set the batch size as eight and the temperature as 0.1. There are also some hyper-parameters in our model and we set $\lambda = 0.1$, $D_{max} = 0.1$, and $W_{min} = 3$.

5 Experimental results

5.1 Overall result

We compare our model with three baselines: QDMiner, QF-I/QF-J. Experimental results are shown in Tables 5 and 6.

(1) First, we compare our model with QDMiner for the accuracy evaluation metrics and the quality of clustering evaluation metrics. As shown in Table 5, our model DeepQFM outperforms QDMiner on both UserQ and RandQ in recall, purity, RI, F1, and F5. For the recall score, DeepQFM achieves 0.355 which exceeds QDMiner's 0.227 by 0.128 on UserQ. For cluster quality evaluation metrics, DeepQFM has achieved results that surpass QDMiner on all clustering evaluation metrics with a little drop on NMI. These

Table 5 Compared with QDMiner on UserQ and RandQ

Dataset	Metric	Pre	Rec	Purity	RI	F1	F5	NMI
UserQ	QDMiner	**0.528**	0.227	0.935	0.772	0.444	0.327	**0.658**
	DeepQFM	0.466	**0.355**	**0.941**	**0.774**	**0.477**	**0.357**	0.653
RandQ	QDMiner	**0.439**	0.193	0.854	0.677	0.526	0.445	**0.504**
	DeepQFM	0.378	**0.340**	**0.893**	**0.699**	**0.578**	**0.492**	0.498

The best performance are given in bold

Table 6 Compared with QDMiner and QF-I/QF-J on UserQ and RandQ.

Dataset	Metric	nDCG	fp-nDCG	rp-nDCG	PRF	wPRF
UserQ	QDMiner	0.618	0.570	0.168	0.351	0.409
	QF-I	0.589	0.334	0.136	0.403	0.431
	QF-J	0.533	0.383	0.119	0.315	0.336
	DeepQFM	**0.634**	**0.573**	**0.203**	**0.425**	**0.543**
RandQ	QDMiner	0.544	0.506	0.153	0.321	0.302
	QF-I	**0.579**	0.275	0.138	0.349	0.388
	QF-J	0.511	0.318	0.108	0.223	0.251
	DeepQFM	0.577	**0.525**	**0.205**	**0.410**	**0.430**

The best performance are given in bold

results demonstrate that the semantic features can better measure the distance between items in lists. By combining the semantic features and statistical features, our model can recall more good facet items.

(2) Similar to Dou et al. (2011), we also compare DeepQFM with QDMiner, QF-I/QF-J on the most important evaluation metrics: nDCG, fp-nDCG, rp-nDCG, PRF, and wPRF for fair. As shown in Table 6, DeepQFM outperforms QDMiner on all evaluation metrics on both datasets. More specifically, for the ranking effectiveness evaluation metrics, nDCG is improved to 0.634 from 0.618, and fp-nDCG is improved to 0.573 from 0.570. DeepQFM has increased rp-nDCG to 0.203 compared to QDMiner's 0.168 by about 21%. As for the accuracy-clustering quality evaluation metrics, PRF has achieved 0.425 and wPRF also gets a huge promotion to 0.543. Besides, DeepQFM also outperforms QF-I/QF-J on all evaluation metrics on both datasets except QF-I for nDCG on RandQ. On RandQ, rp-nDCG for DeepQFM improves to 0.205 from QF-I's 0.138 and QF-J's 0.108. These results indicate that our model can recall more facet items and achieve better clustering and ranking quality.

(3) Our model achieves a comparable result with QDMiner for NMI on the two datasets. Although slightly down from QDMiner for NMI, DeepQFM still outperforms QDMiner on other clustering evaluation metrics. This means that our method still has a good performance in clustering. Besides, the precision of DeepQFM decreases from 0.528 to 0.466 on UserQ and from 0.439 to 0.378 on RandQ. It should be underlined that there is a trade-off between precision and recall. The more good items we recall, the more likely we will recall more bad items. Although slightly inferior in precision to QDMiner, our model has achieved a significant performance on recall. For fp-nDCG

Table 7 Performance of different contrastive samplings

Metric	Pre	Rec	NMI	nDCG	fp-nDCG	rp-nDCG	wPRF
DeepQFM	0.378	0.340	**0.498**	**0.577**	**0.525**	**0.205**	**0.430**
w/o IP	0.361	**0.344**	0.485	0.577	0.523	0.202	0.429
IP+LP	**0.380**	0.282	0.486	0.548	0.488	0.156	0.402

The best performance are given in bold

and rp-nDCG, DeepQFM has also got a huge improvement. This means that DeepQFM can recall as many good items as possible and improve the recall and rp-nDCG with a little drop in precision and without reducing fp-nDCG. Moreover, DeepQFM gets a comparable result with QF-I in terms of nDCG which indicates that the ranking quality of our model is similar to QF-I while the accuracy of finding items and clustering quality are superior to QF-I.

In the following section, we only show the results on RandQ on account of the space limitation. We achieve the same results on RandQ and UserQ in most experiments.

5.2 Ablation study

5.2.1 Effects of contrastive samplings

In this section, we conduct an ablation study to verify the necessity of our contrastive learning tasks. Specifically, we explore the role of item-contrastive sampling (IP) and list-contrastive sampling (LP) and results are shown in Table 7. The row labeled "*w/o* IP" is removing item-contrastive sampling. The row labeled "IP+LP" adopts both item-contrastive sampling and list-contrastive sampling.

The removal of task IP affects precision, NMI and rp-nDCG. This indicates that the item-contrastive sampling task can help the item encoder model the item representation more accurately. Besides, we also conduct experiments to show the effectiveness of list self-contrastive learning. Interestingly, only precision has increased with a huge decrease in other evaluation metrics. Two possible explanations for this are that (1) list-contrastive sampling is self-supervised without human labels; (2) pretraining list representation on sampled list pairs may hurt the semantic information originally contained in BERT. In the future, we plan to design reasonable list-contrastive sampling approaches to learn a better list representation.

5.2.2 Effects of different types of patterns

We add repeat context patterns in Sect. 3.1, so in this section, we conduct an ablation study to verify the effectiveness of repeat context patterns. Experimental results are shown in Table 8, where *w/o* pattern is the result without repeat context patterns. We use the

Table 8 Effects of repeat context pattern, list encoder and item encoder

Metric	Pre	Rec	NMI	nDCG	fp-nDCG	rp-nDCG	wPRF
DeepQFM	0.378	**0.340**	0.498	**0.577**	**0.525**	**0.205**	**0.430**
w/o pattern	0.365	0.335	0.504	0.572	0.516	0.201	0.423
w/o list encoder	0.359	0.185	0.417	0.422	0.384	0.128	0.285
w/o item encoder	**0.387**	0.219	**0.516**	0.575	0.522	0.160	0.326

The best performance are given in bold

remaining three patterns to extract lists from search result documents while other steps follow the same routine with DeepQFM.

By analyzing the results, we find that repeat context patterns are useful for query facets mining because without repeat context patterns, almost all evaluation metrics drop. This means that repeat context patterns can help us find more facet items from search result documents and improve the quality of results. Meanwhile, repeat context patterns also introduce some bad lists and bad items which will lead to lower NMI. For example, for query *watches*, we have extracted list: *camel active watches, casio watches, citizen watches* and with repeat context patterns, we can obtain list: *camel active, casio, citizen*. However, *camel active* is not a facet item and should be removed.

5.2.3 Effects of different components

We further investigate the effects of the list encoder and item encoder in list clustering and item ranking, respectively. Results are shown in Table 8. Without the list encoder, we leverage only the statistical similarity to calculate the distance between lists which harms the results on all evaluation metrics. This indicates that deep semantic features are important to model the relationship between lists. Meanwhile, the removal of the item encoder also causes an impact on recall and rp-nDCG which shows the use of deep semantic features can help our model recall more items and rank facet items higher. It should be noted that we adopt the item ranking approach in QDMiner to conduct the *w/o* item encoder experiment.

5.3 Experiments with clustering distances

We combine deep semantic features and statistical features to provide a better matching score for list clustering. In this section, we investigate the clustering hyperparameter, λ, to illustrate how the performance changes with the integration of deep semantic features. We also wonder in what proportion should deep semantic features and statistical features be combined for the best results. We conduct experiments with the different λ, and experimental results are shown in Fig. 3.

We choose recall, RI, NMI, nDCG, fp-nDCG, rp-nDCG, and PRF to show the influence of λ because they evaluate our model in all aspects. The metrics nDCG, fp-nDCG, rp-nDCG and PRF, are maximized at $\lambda = 0.1$ because at this point, we add statistical information to calculating the clustering distance, compared to $\lambda = 0.0$ which only uses semantic features. As for values of λ greater than 0.1, with the increase of statistical information, deep semantic information gradually decreases, leading to a gradual decline

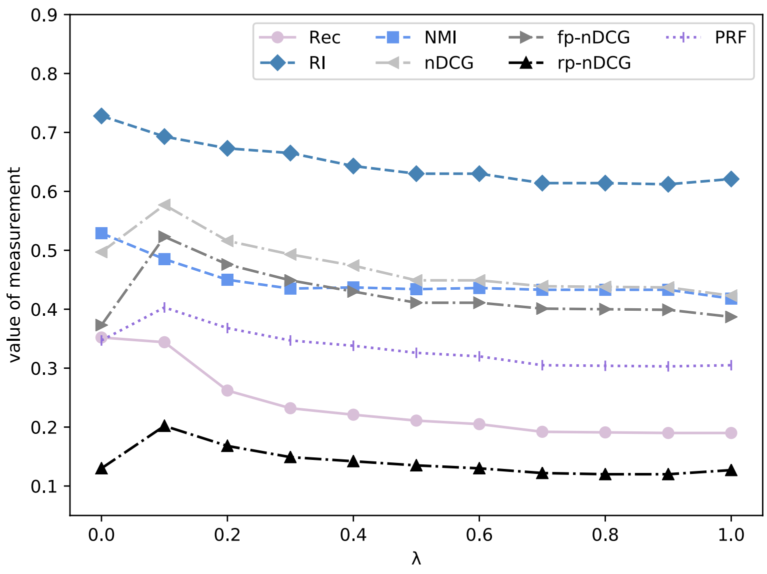

Fig. 3 Experiment result with different value of clustering hyperparameter: λ

Table 9 Test time per query (s) of DeepQFM and QDMiner

Steps	DeepQFM	QDMiner
List extraction	0.220	0.218
List weighting	2.442	2.442
Deep semantic encoder	1.751	–
List clustering	0.766	0.765
Facet ranking and item ranking	0.749	0.005

in performance. This indicates that integrating deep semantic information and statistical information improves results and deep semantic information is important for query facets mining. Additionally, recall, RI, and NMI obtain the best result at $\lambda = 0.0$ and as λ increases, the performance gradually deteriorates. This demonstrates that the more semantic information used, the better the results. However, to obtain a more balanced result, we choose $\lambda = 0.1$ as the hyper-parameter. In short, deep semantic features carry more information, and therefore are helpful to the model.

5.4 Efficiency analysis

In this section, we evaluate the efficiency of our model with QDMiner (Dou et al., 2011). All experiments are conducted on a single TITAN V GPU.

As shown in Table 9, while the model achieves better results, particularly for recall, it comes at the cost of a small increase in processing time. The main time increase lies in the deep semantic encoder and the item ranking, because we need to encode the item and the list and perform deep semantic matching between different items and different lists. In

Table 10 Case study on DeepQFM and QDMiner

Query	Method	Query facets
Watches	QDMiner	rolex, cartier, breitling, omega, tag heuer,..., men,...
		Men, women, unisex, kids, all, children,...
	DeepQFM	breitling, bvlgari, omega, rolex, christian dior, cellini,...
		Women, men, unisex, children, kids,...

an online service, we can store the representation of frequently used items and lists which would reduce the processing time. Besides, the encoding of items and lists in our model can be calculated in parallel, so the actual time of our model is further improved.

5.5 Case study

We present the results of DeepQDM and QDMiner on query *watches* in Table 10 to better understand why our model performs well. Compared with QDMiner, our method DeepQDM leverages deep semantic features which will not cluster items such as *rolex* and *men* together, as they are far away in deep semantic space. Besides, with deep semantic features, useless items like *all*, which often appear in the lists, will not be extracted.

6 Conclusion

In this paper, we study the problem of query facet mining. Existing approaches such as QDMiner, QF-I/QF-J mainly used statistical features to model the representation of lists and items, while some semantic information is also important for mining more good items. To shed light on this research question, we proposed DeepQFM which leverages semantic information to obtain better representations for lists and items and promote the performance of query facets mining. Specifically, we first extract lists from search result documents and assign weights to these lists. Then, we cluster these lists based on their deep semantic representation and statistical features. Finally, we design a contrastive learning task to pretrain an item encoder and remove unimportant items. Experimental results show that our method significantly improves the performance of query facets mining, especially the recall of facet items.

Acknowledgements Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 62272467 and No. 61832017, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, China Unicom Innovation Ecological Cooperation Plan, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. We also wish to acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of RUC.

Author contributions Zhirui Deng and Zhicheng Dou develop the research idea, gather the dataset used in the study, and revise the manuscript. Zhirui Deng designs the research methodology, prepares the experimental datasets, conducts experiments, evaluates the proposed algorithms and drafts the manuscript. J-RW provides financial support and the experiment environment for the research.

Data availability The two datasets we use are available on http://playbigdata.ruc.edu.cn/dou/qdminer/UserQ/ and http://playbigdata.ruc.edu.cn/dou/qdminer/RandQ/.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Aliannejadi, M., Zamani, H., Crestani, F., & Croft, W. B. (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* (pp. 475–484).

Anick, P. (2003). Using terminological feedback for web search refinement: A log-based study. In *Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 88–95).

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004). Query recommendation using query logs in search engines. *International conference on extending database technology* (pp. 588–596).

Basu Roy, S.,Wang, H., Das, G., Nambiar, U., & Mohania, M. (2008). Minimumeffort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM conference on information and knowledge management* (pp. 13–22).

Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., & Yogev, S. (2008). Beyond basic faceted search. In *Proceedings of the 2008 international conference on web search and data mining* (pp. 33–44).

Bing, L., Lam, W., Wong, T.-L., & Jameel, S. (2015). Web query reformulation via joint modeling of latent topic dependency and term context. *ACM Transactions on Information Systems (TOIS), 33*(2), 1–38.

Dakka, W., Dayal, R., & Ipeirotis, P. G. (2006). Automatic discovery of useful facet terms. In *SIGIR faceted search workshop* (pp. 18–22).

Dakka, W., & Ipeirotis, P. G. (2008). Automatic extraction of useful facet hierarchies from text databases. In *2008 IEEE 24th international conference on data engineering* (pp. 466–475).

Dakka, W., Ipeirotis, P. G., & Wood, K. R. (2005). Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM international conference on information and knowledge management* (pp. 768–775).

Damova, M., & Koychev, I. (2010). Query-based summarization: A survey.

Dash, D., Rao, J., Megiddo, N., Ailamaki, A., & Lohman, G. (2008). Dynamic faceted search for discovery-driven analysis. In *Proceedings of the 17th ACM conference on information and knowledge management* (pp. 3–12).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805

Diao, M., Mukherjea, S., Rajput, N., & Srivastava, K. (2010). Faceted search and browsing of audio content on spoken web. In *Proceedings of the 19th ACM international conference on information and knowledge management* (pp. 1029–1038).

Dou, Z., Hu, S., Luo, Y., Song, R., & Wen, J.-R. (2011). Finding dimensions for queries. In *Proceedings of the 20th ACM international conference on information and knowledge management* (pp. 1311–1320).

Dou, Z., Jiang, Z., Hu, S., Wen, J.-R., & Song, R. (2015). Automatically mining facets for queries from their search results. *IEEE Transactions on Knowledge and Data Engineering, 28*(2), 385–397.

Dou, Z., Song, R., & Wen, J. (2007). A large-scale evaluation and analysis of personalized search strategies. In Williamson, C. L., Zurko, M. E., Patel-Schneider, P. F., & Shenoy, P. J. (eds.) *Proceedings of the 16th international conference on world wide web, WWW 2007, Banff, Alberta, Canada*, May 8–12, 2007 (pp. 581–590). ACM. Retrieved from https://doi.org/10.1145/1242572.1242651

Gao, T., Yao, X., & Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. In Moens, M., Huang, X., Specia, L., & Yih, S. W. (eds.) *Proceedings of the 2021 conference on empirical methods in natural language processing, EMNLP 2021, virtual event/Punta Cana, Dominican Republic*, 7–11 November, 2021 (pp. 6894–6910). Association for Computational Linguistics. Retrieved from https://doi.org/10.18653/v1/2021.emnlp-main.552

Gholamrezazadeh, S., Salehi, M. A., & Gholamzadeh, B. (2009). A comprehensive survey on text summarization systems. In *2009 2nd international conference on computer science and its applications* (pp. 1–6).

Herdagdelen, A., Ciaramita, M., Mahler, D., Holmqvist, M., Hall, K., Riezler, S., & Alfonseca, E. (2010). Generalized syntactic and semantic models of query reformulation. In *Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval* (pp. 283–290).

Hu, S., Dou, Z., Wang, X., Sakai, T., & Wen, J.-R. (2015). Search result diversification based on hierarchical intents. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 63–72).

Hu, Y., Qian, Y., Li, H., Jiang, D., Pei, J., & Zheng, Q. (2012). Mining query subtopics from search log data. In Hersh, W. R., Callan, J., Maarek, Y., & Sanderson, M. (eds.) The 35th international ACM SIGIR conference on research and development in information retrieval, SIGIR '12, Portland, or, USA, August 12–16, 2012 (pp. 305–314). ACM. Retrieved from https://doi.org/10.1145/2348283.2348327

Huang, J., & Efthimiadis, E. N. (2009). Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on information and knowledge management* (pp. 77–86).

Jansen, B. J., Spink, A., & Saracevic, T. (2000). Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management, 36*(2), 207–227. https://doi.org/10.1016/S0306-4573(99)00056-4

Jiang, Z., Dou, Z., & Wen, J.-R. (2016). Generating query facets using knowledge bases. *IEEE Transactions on Knowledge and Data Engineering, 29*(2), 315–329.

Jones, R., & Klinkner, K. L. (2008). Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In Shanahan, J. G., et al. (eds.) *Proceedings of the 17th ACM conference on information and knowledge management, CIKM 2008, Napa Valley, California, USA*, October 26–30, 2008 (pp. 699–708). ACM. Retrieved from https://doi.org/10.1145/1458082.1458176

Kong, W., & Allan, J. (2013). Extracting query facets from search results. In *Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval* (pp. 93–102).

Latha, K., Veni, K. R., & Rajaram, R. (2010). AFGF: An automatic facet generation framework for document retrieval. In *2010 International conference on advances in computer engineering* (pp. 110–114).

Li, B., Liu, T., Du, X., Zhang, D., & Zhao, Z. (2015). Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. arXiv:1512.08183

Li, C., Yan, N., Roy, S. B., Lisham, L., & Das, G. (2010). Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of the 19th international conference on world wide web* (pp. 651–660).

Li, L., Zhong, L., Yang, Z., & Kitsuregawa, M. (2013). Qubic: An adaptive approach to query-based recommendation. *Journal of Intelligent Information Systems, 40*(3), 555–587.

Liu, L., Xu, W., Song, W., Du, C., et al. (2015). Query subtopic mining by combining multiple semantics. *International Journal of Multimedia and Ubiquitous Engineering, 10*(12), 341–354.

Maxwell, D., Azzopardi, L., & Moshfeghi, Y. (2017). A study of snippet length and informativeness: Behaviour, performance and user experience. In Kando, N., Sakai, T., Joho, H., Li, H., de Vries, A. P., & White, R. W. (eds.) *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, Shinjuku, Tokyo, Japan*, August 7–11, 2017 (pp. 135–144). ACM. Retrieved from https://doi.org/10.1145/3077136.3080824

Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL conference on computational natural language learning* (pp. 51–61).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv:1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. arXiv:1310.4546

Mitra, M., Singhal, A., & Buckley, C. (1998). Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 206–214).

Niccolai, J. (2009). Yahoo vows death to the '10 blue links'.

Pound, J., Paparizos, S., & Tsaparas, P. (2011). Facet discovery for structured web search: A query-log mining approach. In *Proceedings of the 2011 ACM SIGMOD international conference on management of data* (pp. 169–180).

Radlinski, F., Szummer, M., & Craswell, N. (2010). Inferring query intent from reformulations and clicks. In Rappa, M., Jones, P., Freire, J., & Chakrabarti, S. (eds.) *Proceedings of the 19th international conference on world wide web, WWW 2010, Raleigh, North Carolina, USA*, April 26–30, 2010 (pp. 1171–1172). ACM. Retrieved from https://doi.org/10.1145/1772690.1772859

Riezler, S., Liu, Y., & Vasserman, A. (2008). Translating queries into snippets for improved query expansion. In *Proceedings of the 22nd international conference on computational linguistics (Coling 2008)* (pp. 737–744).

Sarwar, S. M., Addanki, R., Montazeralghaem, A., Pal, S., & Allan, J. (2020). Search result diversification with guarantee of topic proportionality. In Balog, K., Setty, V., Lioma, C., Liu, Y., Zhang, M. & Berberich, K. (eds.) *ICTIR '20: The 2020 ACM SIGIR international conference on the theory of information retrieval, virtual event, Norway*, September 14–17, 2020 (pp. 53–60). ACM. Retrieved from https://doi.org/10.1145/3409256.3409839

Sarwar, S. M., Moraes, F., Jiang, J., & Allan, J. (2021). Utility of missing concepts in query-biased summarization. In Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., & Sakai, T. (eds.) *SIGIR '21: The 44th international ACM SIGIR conference on research and development in information retrieval, virtual event, Canada, July 11–15, 2021* (pp. 2056–2060). ACM. Retrieved from https://doi.org/10.1145/3404835.3463121

Shum, H. (2011). Bing dialog model: intent, knowledge and user interaction. In *Proceedings of the fourth ACM international conference on web search and data mining* (pp. 115—116).

Silverstein, C., Henzinger, M. R., Marais, H., & Moricz, M. (1999). Analysis of a very large web search engine query log. *SIGIR Forum, 33*(1), 6–12. https://doi.org/10.1145/331403.331405

Song, R., Luo, Z., Wen, J., Yu, Y., & Hon, H. (2007). Identifying ambiguous queries in web search. In Williamson, C. L., Zurko, M. E., Patel- Schneider, P. F., & Shenoy, P. J. (eds.) *Proceedings of the 16th international conference on world wide web, WWW 2007, Banff, Alberta, Canada*, May 8–12, 2007 (pp. 1169–1170). ACM. Retrieved from https://doi.org/10.1145/1242572.1242749

Stoica, E., Hearst, M. A., & Richardson, M. (2007). Automating creation of hierarchical faceted metadata structures. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference* (pp. 244–251).

Strohmaier, M., Kröll, M., & Körner, C. (2009). Intentional query suggestion: making user goals more explicit during search. In Craswell, N., Jones, R., Dupret, G., & Viegas, E. (eds.) *Proceedings of the 2009 workshop on web search click data, wscd@wsdm 2009, Barcelona, Spain*, February 9, 2009 (pp. 68–74). ACM. Retrieved from https://doi.org/10.1145/1507509.1507520

Sun, T., Shao, Y., Qiu, X., Guo, Q., Hu, Y., Huang, X., & Zhang, Z. (2020). Colake: Contextualized language and knowledge embedding. arXiv:2010.00309

Surya, D., Deepak, G., & Santhanavijayan, A. (2021). QFRDBF: Query facet recommendation using knowledge centric DBSCAN and firefly optimization. In *International conference on digital technologies and applications* (pp. 801–811).

Szpektor, I., Gionis, A., & Maarek, Y. (2011). Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on world wide web* (pp. 47–56).

Xue, X., & Croft, W. B. (2013). Modeling reformulation using query distributions. *ACM Transactions on Information Systems (TOIS), 31*(2), 1–34.

Zhang, Z., & Nasraoui, O. (2006). Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on world wide web* (pp. 1039–1040).

Zhao, Z., Liu, T., Li, S., Li, B., & Du, X. (2017). Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 244–253).