future internet

MDPI

Article

Query Recommendation Using Hybrid Query Relevance

Jialu Xu †,* and Feiyue Ye †,*

The School of computer engineering and Science, Shanghai University, Shanghai 200444, China

* Correspondence: jialuxu@shu.edu.cn (J.X.); yefy@shu.edu.cn (F.Y.)

† These authors contributed equally to this work.

Received: 22 October 2018; Accepted: 17 November 2018; Published: 19 November 2018

Abstract: With the explosion of web information, search engines have become main tools in information retrieval. However, most queries submitted in web search are ambiguous and multifaceted. Understanding the queries and mining query intention is critical for search engines. In this paper, we present a novel query recommendation algorithm by combining query information and URL information which can get wide and accurate query relevance. The calculation of query relevance is based on query information by query co-concurrence and query embedding vector. Adding the ranking to query-URL pairs can calculate the strength between query and URL more precisely. Empirical experiments are performed based on AOL log. The results demonstrate the effectiveness of our proposed query recommendation algorithm, which achieves superior performance compared to other algorithms.

Keywords: query recommendation; query relevance; query embedding

1. Introduction

As the number of web pages keeps expanding, it is progressively difficult to get useful information which can satisfy user's needs based on original search queries [1]. Thus, users rebuild a new query that is similar to the original search query and is closer to the user's search intentions. We can see some examples in Table 1. For example, when users input a new query "apple" to a website, they do not get their useful information. Thus, the search engine will provide a series of new queries, e.g., "apple website" and, "iPhone". In such a way, users can choose a new query to search relevant information and quickly get what they want. Input queries are usually too short and ambiguous to express the true idea. So, understanding the query and mining intention are the key steps.

Table 1. Example of query recommendation.

Query	Recommended Queries
apple	apple website, apple hk, iphone xr, apple watch4
book	table, pencil box, study, desk
basketball	basketball player, basketball court, soccer

A query log [2] is an important resource to mine user search behavior. The user submits a query to a search engine that leads to a series of information in the query log. The sequences of queries issued by a user within a short time have same intention. A session is defined as sequences of queries that are submitted to satisfy the same intention. Therefore, query co-occurrence in the same session has query relevance and can be used to produce a recommendation. Only using query co-occurrence is prone to data sparsity and loss of much useful information in a query log.

Clicked URLs in a query log have been used in query recommendation [3]. Clicking behaviors show query intention, to some extent. For instance, when the user submits the initial query "apple" to the search engine, its search aims to find the "iPhone official website". If the URL of "Apple's official offer" clicked by the user is considered in the query recommendation, the next recommended query will have the information of iPhone, which is closer to the user's real search intention. Accordingly, the clicked URLs can reveal the user's search intention. Query semantics is also an important factor to understand queries. Both query information and URL information are based on counting the number, and lacking query semantics. To better understand query intention, we propose a query recommendation method. The model is shown in Figure 1.

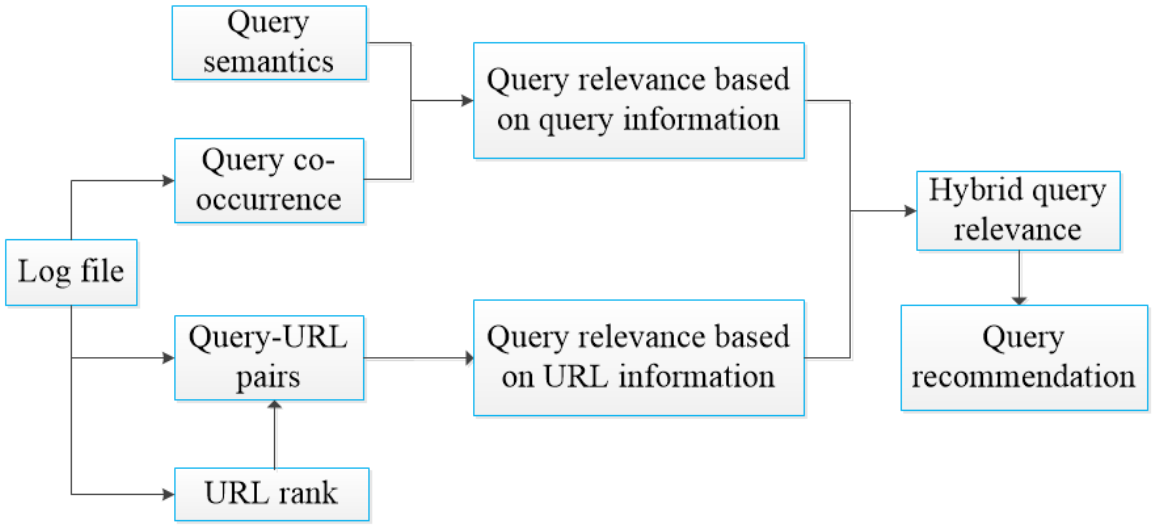


Figure 1. Architecture showing query recommendation.

In our model, we mine query co-occurrence from query log and use query semantics to calculate query relevance. At the same time, we calculate the query relevance by query-URL pairs, adding the ranking of URLs. We can obtain hybrid query relevance, combining the relevance of query information and URL information.

The three major contributions of this paper are summarized as follows:

(1) Solely mining query information from a query log can obtain little useful information and cause data sparsity. Therefore, we use the corpus to train a query embedding vector, getting query semantics to expand the query information and improving the accuracy of the relevance between queries.

(2) We combine the number of clicked URLs and the ranking of URLs in the web pages to calculate query relevance. The two different queries are more similar when they have the numbers of the same clicked URLs. At the same time, the ranking of the URL in the web page is higher; the URL is more related to the query.

(3) We calculate the hybrid query relevance by query information and URL information. Queries in a session have same query intention. The clicked URLs can more accurately understand query intention. Comprehensive consideration of the query information and query-URL pairs is an effective way to understand the user's intention.

2. Related Work

Much research has explored the area of query recommendation based on query logs [4–6]. Chen et al. [4] proposed a query suggestion method by constructing struggling flow graph to identify the struggle phrases and mine effective representation based on query log. Zahera et al. [6] proposed a

method based on clustering processes in which groups of semantically similar queries are detected. A query log has lots of queries which can be used to understand query search intention. Mining query information in a query log has been discussed. Boldi et al. [7] presented query flow graph model (QFG). In a QFG, each node represents a query and each edge between two nodes shows that they are consecutive in a session. Assignment of score values to each query permits use of random walk. It extracts the relationship between queries. However, it has some limitations. On the one hand, only query information is employed in the QFG, but not the URL information and semantic information. The clicked URL in the log and query semantic information can better understand the query semantics and more accurately locate the user's query intention. On the other hand, a query is usually short, with an average of only two or three words. Moreover, some queries are ambiguous. Understanding the semantics of a query and the search intention is limited by the use of only query information in the QFG. Sordoni et al. [8] proposed novel hierarchical recurrent encoder-decoder architecture to account for sequences of previous queries. The queries in a session is training data, thus making the next query prediction contextual. Among these methods presented, some capture word-level representation [9,10], some described queries using different feature space [11], some learned the ranking to improve the accuracy of candidate queries [6].

Clicked URLs are important features to understand query intention. QUBIC [12] was proposed based on a query-URL bipartite graph. It extracts an affinity graph from the initial query-URL bipartite graph only using queries. The weights of edges in an affinity graph are calculated by a query-URL vector, capturing the similarity from query to query. A clustering algorithm [5] was proposed that can automatically mine query major subtopics from the query log, where each subtopic is represented by a cluster containing several URLs and keywords. Nevertheless, these pieces of research are about query-URL pairs for the query recommendation, and are not related to the URL ranking. Ma et al. [13] applied a union matrix which combines query-URL bipartite graph and user-query bipartite graph to learn low-dimensional latent feature vectors of queries and proposed a solution for calculating query similarity using those feature vectors. The query-product clickthrough bipartite graph [14] was proposed by search engine logs and specific domain features such as categories and products popularities. In those approaches above, mining URL information and features can gain query relevance. However, queries submitted to search engines also show the relation between queries. Ye et al. [15] proposed an efficient query suggestion method by calculating the bidirectional transition probability-based query-URL graph and making a strength metric of the query-URL edges. The query log is regarded as the main data in those approaches. However, log files are usually sparse, and there are no edges between many queries and URLs. Therefore, this is not enough for mining query relevance, which only uses a query log.

Existing work aimed to model query information or query-URL pairs to calculate the query relevance, while our method combines query co-occurrence and query semantics to calculate the relevance-based query information and combine the ranking of URL and query-URL pairs to calculate the relevance-based URL information. At the same time, the query information and URL information are considered to calculate the final query relevance.

3. Our Approach

In this section, we illustrate how to calculate the query relevance and recommend the related queries based on query information and URL information. Query relevance based on query information has two parts, namely query co-occurrence in a session and query semantics. The query relevance based on URL information is calculated by query-URL pairs adding ranking. The query recommendation algorithm is based on hybrid query relevance which combines query relevance-based query information and query relevance based on URL information.

3.1. Preliminaries

The users submit a query to the engine and click the returned pages. If users feel satisfied with the information, the search process ends; otherwise, the users submit a new query which has the same search intention as the initial query. Search engine records search behaviors to form a query log. A query log contains a UseID, issued queries, clicked URLs, the ranking of URLs and a timestamp. We can extract useful knowledge to improve the efficiency of query recommendation from this information. A format of a record in a query log, typically, is $< user_{id}, query, clicked \quad URL, ranking, timestamp >$.

A session means that user has a search intention. We consider a query session as a sequence of queries $S = \{q_1, q_2, \ldots, q_n\}$ where n is the number of queries in S. One common way to gain the session from query log is to use a time threshold. We take 30 min as the time threshold for session segmentation according to previous work [16]. White et al. [16] showed the probability of switching, for sessions of varying length, as measured by the number of queries in the session. It can be proved that 30 min is the best threshold for the session partition of the search log.

3.2. Query Relevance Based on Query Information

The query log in the search engine can be divided into different sessions. The queries in a session have the same query intention. We count the number of queries q_i and query q_j in the same session and q_j submitted immediately after the query q_j. We define the query pairs as a tuple, $[q_i, q_j, f(q_i, q_j)]$. So the query log contains many query pairs. The query relevance can be determined by the following equations [7]:

$$Rel_{session}(q_i, q_j) = \begin{cases} \frac{f(q_i,q_j)}{f(q_i)} & if(Rel_{session}(q_i, q_j)) > \theta \\ 0 & otherwise \end{cases} \quad (1)$$

where $Rel_{session}(q_i, q_j)$ denotes the relevance between query q_i and query q_j based on query co-occurrence in sessions. $f(q_i)$ denotes the numbers which q_i appears in the query pairs. $f(q_i, q_j)$ denotes the numbers that query pairs of (q_i, q_j) appeared in the query pairs. Due to data sparsity, there are many missing values in calculating $Rel_{session}(q_i, q_j)$. At the same time, we cannot accurately calculate the query relevance if we do not correctly distinguish whether the queries have the same search intention. Therefore, we use query semantics to expand query information by query embedding vectors, better understanding queries. Word2vec is a good way to train word vectors. The learning process of a vector by word2vec can be expressed as linear translations. For example, we can find the results of simply computing vector ("King") − vector ("Man") + vector("Woman") is very close to the vector of "Queen" [17,18]. Therefore, taking the element-wise sum or mean of the word embedding over all words in the sentence also produces a vector with the potential to encode meaning [19,20]. The queries in the search log are usually short, averaging only two or three words. Therefore, we can get the query embedding vectors based on pre-trained word embedding vector by linear combination. Moreover, the word vector of each word in the query is easily obtained through corpus training. It is a time-saving method.

The calculation of query semantics can be divided into three steps (illustrated in Figure 2):

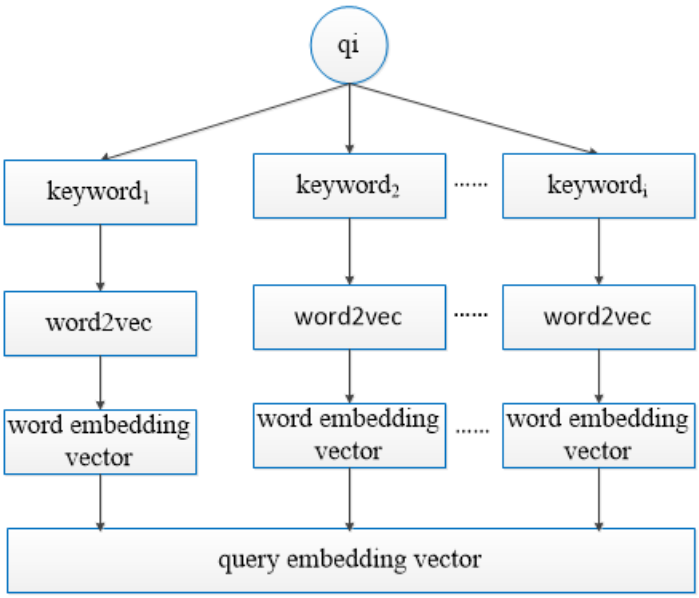


Figure 2. The process of calculating query semantics.

First, each query can be seen as a set of words, represented as $q = \{q_{w1}, q_{w2}, \ldots, q_{wn}\}$, where q is query. The q_{w1} is the keyword in the query.

Second, we calculate the query embedding vectors by pre-trained word embedding vector [19] by the following equations:

$$V_q = \sum_{i=1}^{n} word2vec(q_{wi}) \tag{2}$$

where $word2vec(q_{wi})$ denotes the word i in the query and n denotes the number of words in the query.

Third, we calculate the relevance of query semantics between each query embedding vector via the following formulas:

$$Rel_{sem}(q_i, q_j) = sim(V_{qi}, V_{qj}) = \frac{\sum_{i=1}^{m}(x_i \times x_j)}{\sqrt{\sum_{i=1}^{m}(x_i)^2} \times \sqrt{\sum_{i=1}^{m}(x_j)^2}} \tag{3}$$

where v_{qi},v_{qj} denote the query embedding vector. x_i, x_j denote the value of the query embedding vector v_{qi},v_{qj}. m denotes the dimension of vector.

The query relevance based on query information can be obtained as follows:

$$Rel_{query}(q_i, q_j) = \alpha Rel_{session}(q_i, q_j) + (1 - \alpha) Rel_{sem}(q_i, q_j) \tag{4}$$

where $Rel_{query}(q_i, q_j)$ denotes query relevance based on query information, $Rel_{sem}(q_i, q_j)$ denotes the relevance which calculated by query semantics, $Rel_{session}(q_i, q_j)$ denotes the relevance where queries in same session, and α denotes weight.

3.3. Query Relevance Based on URL Information

Query information is an important factor for understanding query intention. The clicked URLs can also help us better understand query intention. The more the same URLs queries are clicked, the more relevance the queries gets. A higher ranking means that the URL is more important. We count the number of query clicks for each URL and get the average ranking, defined as a tuple, $[Q, URL, C(Q, URL), Ave_{ranking}]$. Q is a set of queries: $Q = \{q_1, q_2, \ldots, q_t\}$. Where t is the number of queries. URL is a set of URLs: $URL = \{u_1, u_2, \ldots, u_h\}$. Where h is the number of URL. $Ave_{ranking}$ is the average ranking of each URL. For query q_i in a set of Q, we can see the structure of query, clicked URLs and ranking in Figure 3.

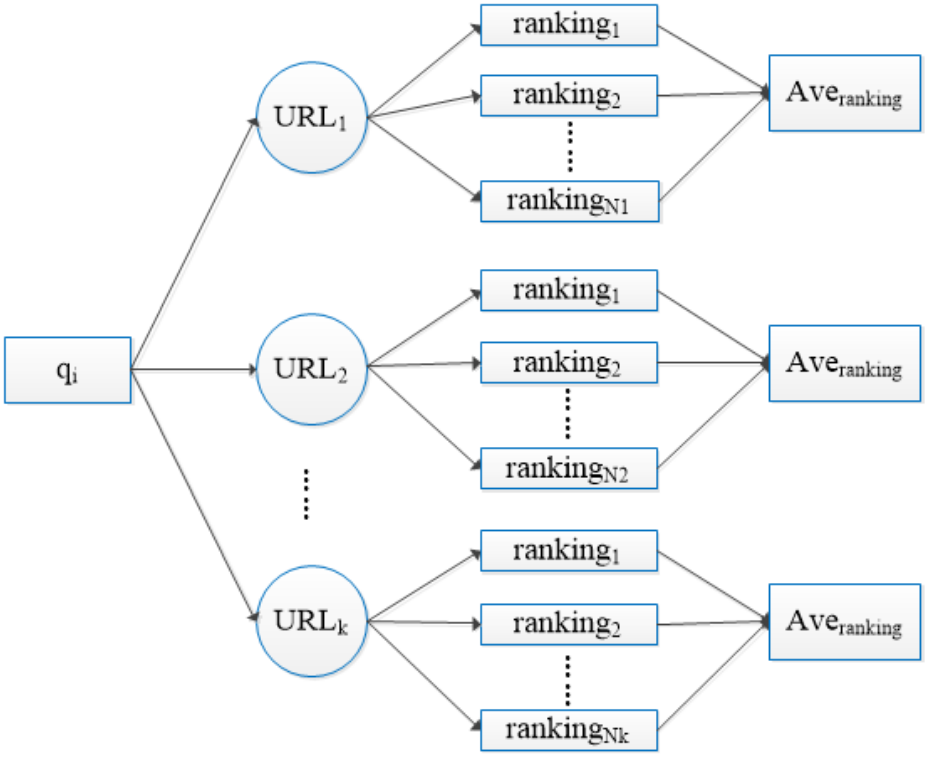


Figure 3. The structure of query, clicked URLs and ranking.

From Figure 3, we can see that query q_i clicks different URLs, and each URL has a different ranking. Therefore, we calculate the average rankings of each URL by Equation (5).

$$Ave_{ranking}(u_i) = \frac{\sum_{i=1}^{|N_i|} R_{u_i}}{|N_i|} \tag{5}$$

where $Ave_{ranking}(u_i)$ denotes the average ranking of URL_i. $|u_i|$ denotes the total number of rankings in URL_i. R_{u_i} denotes one of the rankings when query q_i clicks the URL_i. $|N_i|$ denotes the number of ranking in $|u_i|$.

We combine the number of clicks and the ranking. However, in a query log, the higher ranking means that the URL is in front of the web page and the value of ranking is small. The strength between the query and URL can be calculated as follows:

$$\begin{aligned} Str(q, u_i) &= \frac{C(q, u_i)}{\sum_{i=1}^{|k|} C(q, u_i)} * Ave_{ranking}(u_i) \\ &= \frac{C(q, u_i)}{\sum_{i=1}^{|k|} C(q, u_i)} * \frac{\sum_{i=1}^{|N_i|} R_{u_i}}{|N_i|} \end{aligned} \tag{6}$$

where $C(q, u_i)$ denotes the number that query q clicks URL u_i. $|k|$ denotes the number of URL that query q clicks.

Given the query and URL, we can get a $t \times h$ matrix $S(s_{ij})$ which shows the strength of query and URL. s_{ij} denotes the strength of query and URL. t denotes the number of queries and h is the number of URL.

$$S(s_{ij}) = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1h} \\ s_{21} & s_{22} & \cdots & s_{2h} \\ \cdots & \cdots & \cdots & \cdots \\ s_{t1} & s_{t2} & \cdots & s_{th} \end{bmatrix} \tag{7}$$

The relevance of queries based on URL information can be measured using the cosine measure [21] as follows:

$$\begin{aligned} Rel_{URL}(q_i, q_j) &= \frac{S(s_i) \cdot S(s_j)}{|S(s_i)||S(s_j)|} \\ &= \frac{\sum_{k=1}^{h} (s_{ik} \times s_{jk})}{\sqrt{\sum_{k=1}^{h} (s_{ik})^2} \times \sqrt{\sum_{k=1}^{h} (s_{jk})^2}} \end{aligned} \tag{8}$$

where $S(s_i)$ denotes the ith row of the matrix $S(s_{ij})$, s_{ik} denotes an elements in matrix $S(s_{ij})$, $S(s_j)$ denotes the j_{th} row of the matrix $S(s_{ij})$, s_{jk} denotes an elements in matrix $S(s_{ij})$.

3.4. Hybrid Query Relevance

Not only can query information be used to understand search intentions, but also URL information. However, there still exists drawbacks that make obtaining comprehensive query relevance in depth inefficient. We define a hybrid query relevance which takes advantage of each method as follows:

$$Rel(q_i, q_j) = \beta Rel_{query}(q_i, q_j) + (1 - \beta) Rel_{URL}(q_i, q_j) \tag{9}$$

where $Rel(q_i, q_j)$ denotes the hybrid query relevance, $Rel_{query}(q_i, q_j)$ denotes the query relevance based on query information, $Rel_{URL}(q_i, q_j)$ denotes the query relevance by URL information, and β denotes weight. The contrast experiments will be made to find out the optimum weight for parameter β in the experimental part.

When users input a query into a search engine, we use restart random walk [22] to recommend the query which is close to the input query. Random walk with restart is defined as equation [10].

$$\vec{r_i} = c\widetilde{W}\vec{r_i} + (1 - c)\vec{e_i} \tag{10}$$

where c is restart probability and $\widetilde{W}$ is the matrix of hybrid query relevance. $\vec{e_i}$ stands for initial vector, The ith element is 1, the rest is 0. $\vec{r_i}$ is scoring vector.

In the process of recommendation, the initial query, as a starting point, randomly selects an adjacent query with the initial query, and moves to the adjacent query. Then the current adjacent query, as the initial, queries and repeats the above process of random walk. Finally, we find the top queries to recommend to users that are similar to the initial query.

4. Results

In the section, we first introduce the data set and evaluation methods. Then we find the appropriate values of parameters α, β by gradually adjusting their weights. Last, we validate the performance of our proposed algorithm through several experiments which compare our algorithm with other algorithms. All the recommendation algorithms are implemented in Python 2.7 version on Windows 10 running on a PC with system configuration Intel Core i5 processor (2.40 GHz) with 8-GB RAM.

4.1. Experimental Data and Evaluation Methods

The data set used in this paper comes from search logs from AOL search engine from March to May in 2006 (http://www.researchpipeline.com/mediawiki/index.php?title=AOL_Search_Query_Logs). This collection consists of approximately 20 million web queries collected from approximately 6.5 million users over three months. We list a few examples from AOL log in Table 2.

Table 2. Examples of AOL log.

UseID	Query Content	Timestamp	URL Ranking	Clicked URL
217	lottery	1 March 2006 11:58:51	1	http://www.calottery.com
1268	ozark horse blankets	1 March 2006 17:39:28	8	http://www.blanketsnmore.com
2334	jesse mccartney	1 March 2006 18:53:50	4	http://www.hyfntrak.com
2421	cstoons	9 May 2006 17:32:44	2	http://www.xtracrispy.com

We use a 10-fold cross validation algorithm. The data set is randomly divided into ten parts. Each copy contains approximately 3,500,000 records and 800,000 sessions. We take 9 copies as a training set, and 1 copy as a test set each time. We repeat the experiments 10 times to get the mean value of the results.

The preprocessing of the training set involves three steps: first, we use the threshold of 30 min to divide the sessions to estimate whether the two queries have the same search target. Subsequently, www and other navigation vocabulary in the query are removed, which can reduce noise. Finally, we remove the edges between the queries less than five times and the edges between the queries and URL less than five times.

During the test, the total queries submitted after query q in the test set are determined and are considered part of a session with q to form a relevant query set. If the recommended query is in the relevant query set, it is considered successful. In this study, the first N queries are selected to evaluate the precision, recall, and F1-value. The precision, recall, and F1-value are expressed as follows:

$$\text{precision} = \frac{\text{the number of correct queries}}{\text{the number of total queries}} \tag{11}$$

$$\text{Recall} = \frac{\text{the number of correct queries}}{\text{the number of total correct queries}} \tag{12}$$

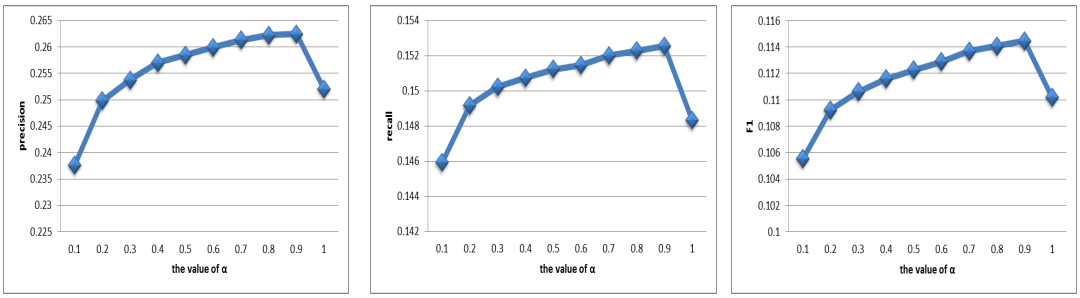
$$\text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{13}$$

4.2. Selection of Parameters α, β

4.2.1. Selection of Parameter α

We do multi-group experiments for the value for parameter α. In the experiments, the parameter α is satisfied at more than 0 and less than or equal to 1, when $\alpha = 1$ means that we do not add the query semantics. We change their value with interval of 0.1 in the experiment and observe the influence of the precision, the recall and the F1 measure. To get accurate results, we recommend queries from Top 5 to Top 50 with interval of 5 and calculate average value to get the final precision, the recall, and the F1 measure. The results are shown in Figure 4.

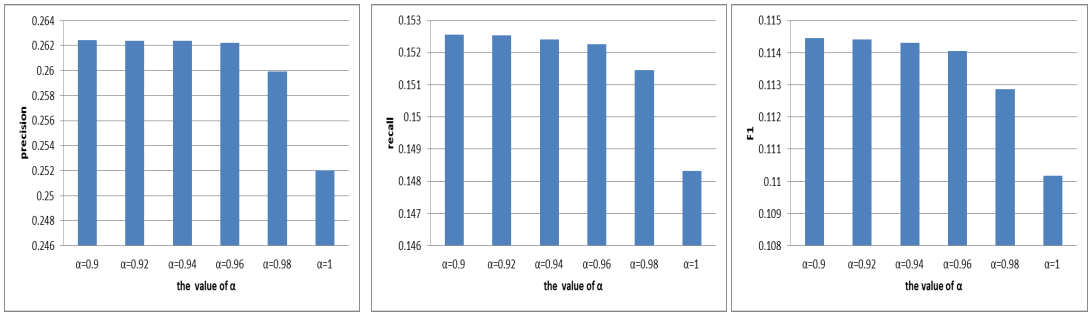
From Figure 4, we can see that combining the query semantics can improve query recommendation results. When the parameter α is too small, that means the weight of query semantics is too big, and we cannot get good results. This is because some queries are ambiguous. At the same time, some queries are correlated, but they do not have semantic relevance. We can mine their relevance by query log. However, only using query pairs ($\alpha = 1$) also cannot get good recommendation results. Because of the sparsity of data and the incorrect session partition, the relevance between many queries is missing. Therefore, we combine query pairs and query semantics. In Figure 4c, we can see that we can get better results compared with only using query pairs when the parameter α is larger than 0.3.



(**a**) Precision. (**b**) Recall. (**c**) F1.

Figure 4. Selection of Parameter α. (**a**) Description of Precision in the first panel. (**b**) Description of recall in the second panel. (**c**) Description of F1 in the third panel.

We can observe that precision, recall, and F1 declined beyond 0.9. To get the optimal values of the α, we change the range of parameter to 0.02 and conduct multiple experiments. The results are shown in Figure 5.



(**a**) Precision. (**b**) Recall. (**c**) F1.

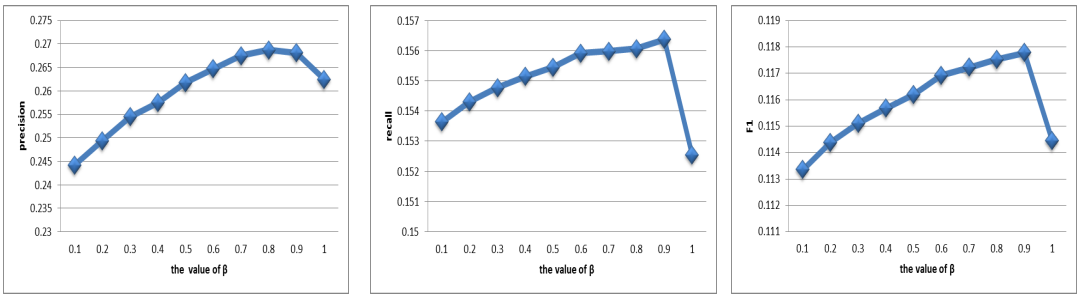
Figure 5. Selection of Parameter α. (**a**) Description of precision in the first panel. (**b**) Description of recall in the second panel. (**c**) Description of F1 in the third panel.

Figure 5 suggests that precision, recall, and F1 decrease with the increase of parameter. $\alpha = 0.9$ is the best result. Thus, we set the parameter α to 0.9 in the later experiments.

4.2.2. Selection of Parameter β

Parameter β is the weight to balance the query information. A large number of experiments have been done by changing their value with interval of 0.1. β is 1, which means we only use query information. We also use the precision, the recall, and the F1 measure to evaluate results. The queries are recommended from Top 5 to Top 50 with intervals of 5. We calculate the average precision, the recall, and the F1 measure. Figure 6 shows the results.

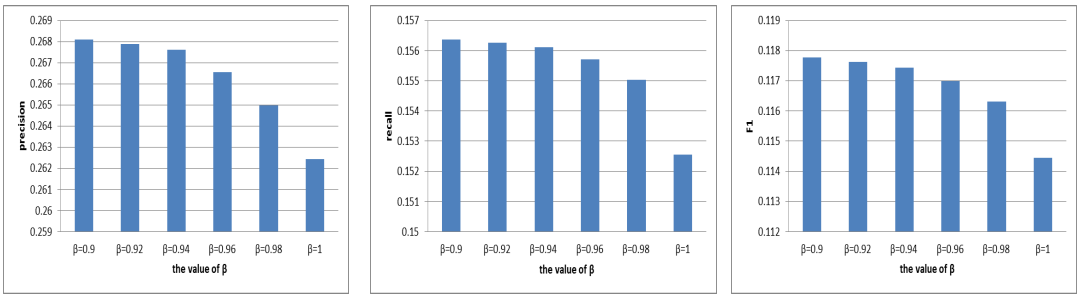
Figure 6a shows that the precision is not greatly improved when we add the URL information. Due to the data noise, the precision is lower than when only using query information sometimes. However, we can see the recall is greatly improved in Figure 6b. URL information can be used as complementary features to better understand user search intention. We can find the relevance of queries more widely by clicked URLs. That is why the recall is greatly improved. Based on an overall consideration of precision and recall, we can see that the recommendation efficiency is improved in Figure 6c.



(**a**) Precision. (**b**) Recall. (**c**) F1.

Figure 6. Selection of Parameter β. (**a**) Description of Precision in the first panel. (**b**) Description of recall in the second panel. (**c**) Description of F1 in the third panel.

To get the optimal values of the β, we also change the range of parameter to 0.02 and conduct multiple experiments. The results are shown in Figure 7.



(**a**) Precision. (**b**) Recall. (**c**) F1.

Figure 7. Selection of Parameter β. (**a**) Description of precision in the first panel. (**b**) Description of recall in the second panel. (**c**) Description of F1 in the third panel.

In Figure 7, we can observe that precision, recall, and F1 declined rapidly beyond 0.9. This is because the query submitted by the user is the best way to reflect the query intention, and the clicked URL can be the supplementary condition to better understand the query intention in the process of query recommendation. Therefore, the weight of query information is relatively large. $\beta = 0.9$ is the best result. Therefore, we set parameter β to 0.9 in the later experiments.

4.3. Evaluation of Efficiency

To examine the effectiveness of our approach, we compare the performance of the following algorithms:

(1) QFG [7]: This is a query flow graph model extracting queries to count the number of query co-occurrences.
(2) QUBIC [12]: This is a bipartite graph model using query information and URL information in logs to build a query-URL bipartite graph.
(3) RW_{UQ} [15]: This is a method calculating the bidirectional transition probability-based query-URL graph and making a strength metric of the query-URL edges.
(4) CQM [6]:This is a method based on clustering processes in which groups of semantically similar queries are detected.
(5) QRSR: Our method considers query information and URL information. The relevance based on URL combines the query-URL pairs with URL ranking which can more accurately calculate the relation between query and URL.

The precision of the different algorithms is shown in Figure 8.

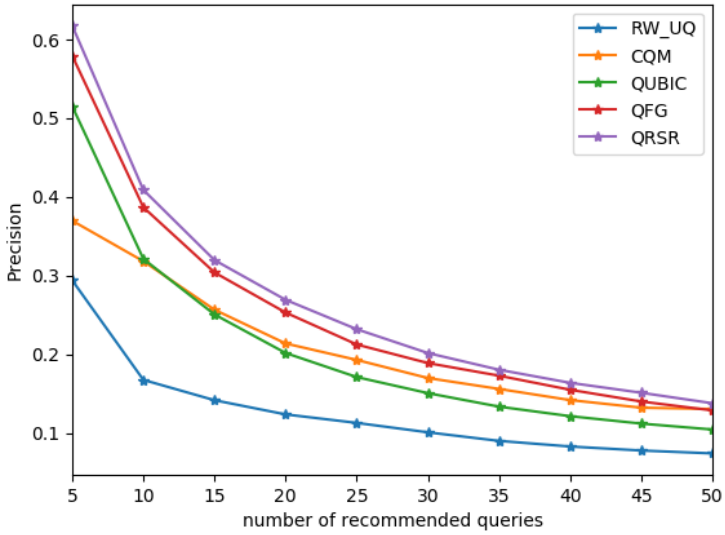
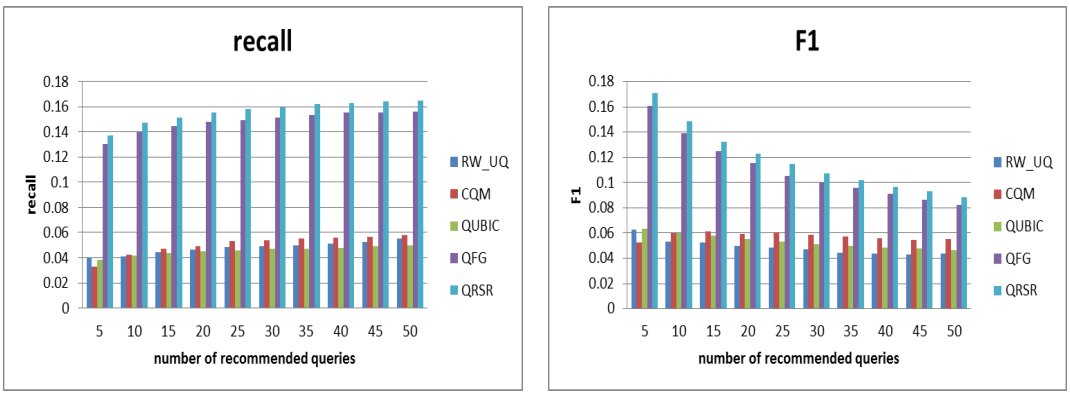


Figure 8. Compare precision with other algorithms.

From Figure 8, we can see that our method has higher precision than other query recommendation algorithms. We cannot get wide and accurate relation of queries solely using query information or URL information. It is limited to understanding query intention. Query information and user behavior information can complement each other. Using query embedding vectors to represent query semantics can better understand query; using the ranking of a URL can improve the accuracy of the strength between query and URL. We combine query information and URL information to mine more relevance between queries which can more accurately understand queries and query intention.

Figure 9 shows the recall and F1-value on the different algorithms, respectively.



(**a**)Precision. (**b**)Recall.

Figure 9. Compare recall and F1 with other algorithms. (**a**) Description of recall in the first panel. (**b**) Description of F1 in the second panel.

In Figure 9a, the recall of our approach is compared with those of the other four methods. We can observe that as the number of recommendations increases, the recall rate of our method as well as other methods increases. However, our method has a higher recall than the other two methods. The results of F1-value are shown in Figure 9b. Its trend is the same as that observed in Figure 9a.

5. Conclusions

In this paper, we presented a query recommendation algorithm to understand search intention by using both query information and URL information. The query semantics was used to calculate query relevance-based query information. Using the ranking of URL can better measure the strength between query and clicked URL. Experiments based on an AOL log suggest that our method has higher precision in query recommendation. In future work, we will mine other information in the search log to improve recommendation results, which can be closer to the query intention.

Author Contributions: J.X. proposed the query embedding and the vector similarity measures; F.Y. gave the URL strength approach and the analysis; all the authors wrote the manuscript and revised the final version.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kop, R. The Unexpected Connection: Serendipity and Human Mediation in Networked Learning. *J. Educ. Technol. Soc.* **2010**, *2*, 2–11.
2. Choudhary, D.; Subhash, C. Adaptive Query Recommendation Techniques for Log Files Mining to Analysis User's Session Pattern. *Int. J. Comput. Appl.* **2016**, *133*, 22–27. [CrossRef]
3. Thirumalai, C.S.; Sree, K.S.; Gannu, H. Analysis of Cost Estimation Function for Facebook Web Click Data. In Proceedings of the IEEE International Conference on Electronics, Communication and Aerospace Technology Iceca, Coimbatore, India, 20–22 April 2017.
4. Chen, Z.; Yamamoto, T.; Tanaka, K. Query Suggestion for Struggling Search by Struggling Flow Graph. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Omaha, NE, USA, 13–16 October 2017; pp. 224–231.
5. Hu, Y.; Qian, Y.; Li, H.; Jiang, D.; Pei, J.; Zheng, Q. Mining query subtopics from search log data. In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, Portland, OR, USA, 12–16 August 2012; pp. 305–314 .
6. Zahera, H.M.; El-Hady, G.F.; El-Wahed, W.F.A. Query Recommendation for Improving Search Engine Results. *Lect. Notes Eng. Comput. Sci.* **2010**, *2186*, 45–52.
7. Boldi, P.; Bonchi, F.; Castillo, C.; Donato, D.; Gionis, A.; Vigna, S. The query-flow graph: Model and applications. In Proceedings of the CIKM'08, Napa Valley, CA, USA, 26–30 October 2008; pp. 609–618.
8. Sordoni, A.; Bengio, Y.; Vahabi, H.; Lioma, C.; Simonsen, J.G.; Nie, J.Y. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In Proceedings of the CIKM'15, Melbourne, Australia, 19–23 October 2015; pp. 553–562.
9. Bonchi, F.; Perego, R.; Silvestri, F.; Vahabi, H.; Venturini, R. Efficient query recommendations in the long tail via center-piece subgraphs. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, Portland, OR, USA, 12–16 August 2012; pp. 224–231.
10. Qiao, D.; Zhang, J.; Wei, Q.; Chen, G. Finding Competitive Keywords from Query Logs to Enhance Search Engine Advertising. *Inf. Manag.* **2016**, *54*, 531–543. [CrossRef]
11. Li, X.; Guo, C.; Chu, W. Deep learning powered in-session contextual ranking using clickthrough data. In *Workshop on Personalization: Methods and Applications, at Neural Information Processing Systems*; Computer Sciences and Statistics: Madison, WI, USA, 2014.
12. Li, L.; Yang, Z.; Liu, L.; Kitsuregawa, M. Query-url bipartite based approach to personalized query recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; pp. 1189–1194.
13. Ma, H.; Yang, H.; King, I.; Lyu, M.R. Learning latent semantic relations from clickthrough data for query suggestion. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, CA, USA, 26–30 October 2008; pp. 709–718.
14. Noce, L.; Gallo, I.; Zamberletti, A. Query and Product Suggestion for Price Comparison Search Engines based on Query-product Click-through Bipartite Graphs. In Proceedings of the International Conference on Web Information Systems and Technologies, Rome, Italy, 23–25 April 2016; pp. 17–24.

15. Ye, F.; Sun, J. Combining Query Ambiguity and Query-URL Strength for Log-Based Query Suggestion. In Proceedings of the International Conference on Swarm Intelligence, Brussels, Belgium, 7–9 September 2016.
16. White, R.W.; Dumais, S.T. Characterizing and predicting search engine switching behavior. In Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China, 2–6 November 2009; pp. 87–96.
17. White, L; Togneri, R.; Liu, W.; Bennamoun, M. How Well Sentence Embeddings Capture Meaning. In Proceedings of the 20th Australasian Document Computing Symposium, Parramatta, Australia, 8–9 December 2015; pp. 1–8.
18. Rong, X. Word2vec Parameter Learning Explained. *Comput. Sci.* **2015**, arXiv:1411.2738.
19. Li, Y.; Lyons, K. Word representation using a deep neural network. In Proceedings of the International Conference on Computer Science and Software Engineerin, Toronto, ON, Canada, 31 October–2 November 2016; pp. 268–279.
20. Ling, W.; Dyer, C.; Black, A.W.; Trancoso, I. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. In Proceedings of the Conference on North American Chapter of the Association for Computational Linguistics—Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015.
21. Barbosa, J.J.G.; Solís, J.F.; Terán-Villanueva, J.D.; Valdés, G.C.; Florencia-Juárez, R.; Mata, M.B. *Mojica: Implementation of an Information Retrieval System Using the Soft Cosine Measure*; Springer International Publishing: Berlin, Germany, 2017.
22. Tong, H.; Faloutsos, C.; Pan, J.Y. Fast Random Walk with Restart and Its Applications. In Proceedings of the International Conference on Data Mining, Hong Kong, China, 18–22 December 2016; pp. 613–622.