Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An incremental model on search engine query recommendation

JianGuo Wang [a,*], Joshua Zhexue Huang [b], Dingming Wu [b], Jiafeng Guo [c], Yanyan Lan [c]

[a] Shenzhen Key Laboratory of High Performance Data Mining, Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, 1068 Xueyuan Avenue, Shenzhen University Town, Shenzhen 518055, PR China

[b] College of Computer Science and Software Engineering, Shenzhen University, 3688 Nanhai Avenue, Shenzhen 518060, PR China

[c] Institute of Computing Technology, Chinese Academy of Sciences, 6 Kexueyuan South Road, Beijing 100190, PR China

A R T I C L E I N F O

Article history:
Received 25 April 2016
Received in revised form
15 July 2016
Accepted 8 September 2016
Communicated by Zidong Wang

Keywords:
Query recommendation
Query log analysis
Convex optimization
Incremental update method

A B S T R A C T

Search engine query recommendation based on mining query logs has been considered as an important and useful method of facilitating users to retrieve information. However, the log data evolves quickly. Existing query recommendation approaches have to rebuild the models when new log data arrive. In this paper, we extend the query ranking model (QRM) proposed in our previous work (Wang et al., 2015) [1] to an adaptive model in which new coming log data is incrementally added, so that the recommendation model is kept up-to-date. The experimental results have demonstrated that the proposed incremental query ranking model (IQRM) is able to recommend queries more efficiently than re-building QRM on evolving log data without losing accuracy.

1. Introduction

Search engines such as Google, Yahoo!, and Bing have become ubiquitous tools for people to retrieve information from the Web. However, it is difficult for users to compose a succinct and precise query to express their information needs. To reduce the users' burden of formulating queries, search engines provide query recommendation services.

In general, query recommendations are usually made by mining query logs. Given a user-specified query q, candidate queries $\{q_1, q_2, \ldots, q_m\}$ for recommendation are ranked based on their similarities w.r.t. q, i.e., $S(q, q_i)$, where function $S()$ can be computed in different ways, e.g., commonly clicked Uniform Resource Locators (URLs) [2–4] and consecutively reformulated queries in the same search session [5–7].

Since the recommendation model is built on query logs previously collected, its effectiveness decreases due to interest shifts [8]. For example, given a user-specified query 'da vinci', the recommendations such as 'last supper da vinci' and 'leonardo da vinci paintings' are out-of-date in May 2006 compared with 'da vinci code film' and 'da vinci code book', because in May 2006 the movie adaptation of the popular book 'Da Vinci Code' was released. To reduce the effect of aging, recommendation model must be periodically re-built to catch up interest shifts. Existing approaches on query recommendation do not provide incrementally updating mechanism. By incorporating new coming log data, the model has to be re-built [9]. In practice, the rebuilding the model is time-consuming. Hence, a feasible query recommendation model should be able to efficiently update itself base on the evolving query logs.

Another big challenge for the current query recommendation approaches lies in processing big data. Commercial search engines receive billions of queries and collect tera bytes of log data everyday [10]. We propose to partition query logs into several blocks and update the recommendation model block by block.

In this paper, we extend the probabilistic query-ranking model (QRM) [11] to an efficient incremental query-ranking model (IQRM) which can incrementally update itself. We develop some novel optimization technologies to accelerate the updating process. Most of the computed information are reused by IQRM. Therefore, IQRM is much more efficient than QRM on processing evolving query log.

The main contributions of this paper are as follows: (1) we extend QRM model to an IQRM model which can process evolving query logs efficiently and keep the recommendation model up to date; (2) since IQRM can process query log block by block, IQRM shows advantages in processing big data; (3) extensive experiments were conducted on a real query log data and the performance of IQRM and comparisons with QRM on evolving query log were analyzed. The execution time of IQRM was about 8% of the execution time of re-running QRM. We also compared the quality

* Corresponding author.
E-mail address: hellowangjg@126.com (J. Wang).

http://dx.doi.org/10.1016/j.neucom.2016.09.003
0925-2312/© 2016 Elsevier B.V. All rights reserved.

of recommendations generated by IQRM and QRM with human judge. The comparison results have shown that the quality of recommendations of IQRM and QRM were equivalent.

The rest of this paper is organized as follows. In Section 2, we give a brief review of related work. in Section 3, we present the batch method for building recommendation query base with QRM. In Section 4, we introduce the incremental method for updating recommendation query base. In Section 5, we present the experiments and the analysis of experimental results. In Section 6, we give some conclusions and discuss the future work.

2. Related work

2.1. Similarity-based query recommendation

Existing query recommendation commonly rank the candidate queries set $\{q_1, q_2, \ldots, q_m\}$ of initial query q by a similarity function $S(q, q_i)$, where S is computed from the query log data of q and q_i. Different log data have been used to compute S.

The clicked URLs in query log data were first used to compute the similarity between two queries. A query-URL bipartite graph was created from the URLs in query log data and then used to compute the similarities between queries. Ref. [2] used an agglomerative clustering algorithm to cluster queries and find relevant queries for recommendation. Ref. [12] applied two types of random walk process to propagate the query similarity along the query-URL bipartite graph and obtained better similarity scores between queries. Ref. [13] transformed the undirected query-URL bipartite graph into a directed one and applied a random walk to find queries similar to the initial query. Instead of random walk, Ref. [4] used heat diffusion to model similar information propagation on the directed query-URL bipartite graph for query recommendation. Ref. [14] represented query with a vector of URLs. Each component of the vector was weighted according to the number of times the corresponding URL had been clicked when returned by that query.

The log data of search sessions were also used to compute the similarity between two queries. A search session is the sequence of queries issued by the same user in a given time period. Ref. [15] formulated search sessions as transactions of queries and applied association rule mining algorithms to find associated queries for recommendation. Ref. [16] represented each query as a vector of search sessions and each search session index recorded the frequency of the query occurs in that session. The similarity between two queries was computed from the two query vectors. Ref. [17] proposed to use a Mixture Variable Memory Markov model built from search sessions to predict the next query to be selected given the current search session.

From the adjacent queries in search sessions, Refs. [5,6] built a query-flow graph and applied a random walk method, starting from the initial query, to find similarities between queries. Ref. [7] proposed a method to perturb the transition probabilities of the query-flow graph to maximize the expected utility of the random walk. Ref. [18] proposed a method to project a large query-flow graph to a low dimension space to reduce the similarity computation between queries.

2.2. Incremental query recommendation

Existing incremental methods were not designed for search engine query recommendation,therefore,they are different from our IQRM. For example, Ref. [19] developed an incremental density-based ensemble clustering over evolving data streams; Ref. [20] proposed an incremental fuzzy neural network that based on meta-cognitive; Ref. [21] introduced an incremental regularized extreme learning machine while Ref. [22] developed a Gaussian mixture framework for incremental nonparametric regression.

Even through there are some literatures on incremental query recommendation, they are also different from ours. Ref. [8] only analyzed the aging effect on the query-flow graph [5] and validated that the recommendations generated from historical log data would go out-of-date. Based on that analysis, Ref. [23] introduced a simple incremental algorithm for updating an existing query-flow graph without explicit analysis on efficiency. Ref. [9] extended an association rule based method used in [15] to an incremental update method. [9] also extended the query recommendation method used in [14] to an incremental update method. Such incremental update method like ours for probabilistic model has not been explicitly addressed.

3. Batch method for updating recommendation query base with QRM

Query-ranking model (QRM) [11] builds a probabilistic model for query recommendation. QRM first reorganizes the query logs in search sessions and then collects the search sessions with the same initial query into a search sessions group. First query in a search session is defined as initial query. Given an initial query q, the set of all distinct queries $\mathbf{Q} = \{q_t | 1 \le t \le |\mathbf{Q}|\}$ in its search sessions group SSG are defined as candidate queries of q, excluding q itself. To conduct recommendations for q, QRM calculates perceived utility α, posterior utility β and query-level utility γ for each candidate query of q, ranks each candidate query with $\alpha_*\beta_*\gamma$ and recommends top k to users.

Specifically, given a candidate query q_t of q, its perceive utility α_t is calculated as the fraction of the frequency that the search results of q_t are clicked in SSG over the total frequency that q_t occurs in SSG. Let $\boldsymbol{\beta}$ be the posterior utilities of $\mathbf{Q}$, QRM obtains them by resolving the optimization problem as follows:

$$\begin{aligned}\text{maximize } \mathcal{F}(\boldsymbol{\beta}) = \sum_{j=1}^{N}\sum_{i=1}^{M_j} & S_{j,i}\cdot\log\left(\sigma\left(\sum_{k=1}^{i} I(C_{j,k}=1)\cdot I(O_{j,k}=1)\cdot\beta_{j,k}\right)\right) \\ &+ (1-S_{j,i})\cdot\log\left(1-\sigma\left(\sum_{k=1}^{i} I(C_{j,k}=1)\cdot I(O_{j,k}=1)\cdot\beta_{j,k}\right)\right) \\ &\text{subject to } \boldsymbol{\beta} \ge \mathbf{0},\end{aligned} \tag{1}$$

where N denotes the number of search sessions in search sessions group SSG, M_j denotes the length of search session j, $S_{j,i}$ denotes the satisfied status at i position in search session j, 1 or 0, $C_{j,k} = 1$ denotes that the search results of query at k position in search session j are clicked, $O_{j,k} = 1$ denotes that the query at k position in search session j is related to initial query q, $\sigma(.)$ denotes the logistic function

$$\sigma(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x})}{1+\exp(\boldsymbol{x})}, \tag{2}$$

$I(.)$ is indicator function and $\beta_{j,k}$ denotes the posterior utility of candidate query at k position in search session j.

Given a search sessions group SSG, QRM computes θ_j for search session j,

$$\theta_j = \sigma\left(\sum_{i=1}^{M_j} I(O_{j,i}=1)\cdot|U_{j,i}|\right), \tag{3}$$