Research Paper Engineering

Organizing User Search Histories into Relevant Query Groups by Using Em – Algorithm

*G.Vijay Kumar ** A. Krishna Chaitanya

* M.Tech (C.S.E), VCE,Hyderabad

** Associate Professor in IT Dept, VCE,Hyderabad

ABSTRACT

Creating search histories is a difficult process in the web. In this paper we are proposing EM algorithm for creating search histories. This algorithm is used to convert the related queries into a group or related searches into a group called as "query group". In this EM algorithm we follow two steps those are Expectation step, Maximization Step, with these two steps it is easy to divide the queries into their related query groups.

1. Introduction:

This paper shows how to organize user's history in the search engine, means if the user search in the search engine, then that user query and URL will be stored in the history log. To reduce the burden on the users, in this paper we have created the "Query Group" feature. In this group we store user queries. This Query Group having only related queries and query groups will be created with different related queries. For example Several users have been searching for banks, then that time a new Query Group will be created about bank and another user have been searching for Hospitals information then a new Query Group will be created automatic and Dynamic fashion. This feature will be useful to the users for selecting or searching related queries.

The users no need to bother about whether they are searching for related data or not. After creating query group, if the user wants to search for the information about the bank, then that page shows navigation to the user's query group then he can choose or select what he required. Here, time will be saved and there no burden on the search engine. If the keyword will match to the existing keywords then that page will be navigate the corresponding Query Group. To create related query group, he has been used Expectation–Maximization (EM) clustering algorithm and three graphs, Query reformulation graph, Click graph and Fusion graph. Recently they used some algorithms like, Select best query group Relevance group Select next node to visit User can get the data what he required, depends on the entering keyword in the search engine. Users can search in the Query Group for which, minimum clicks will be required, and time consumption will be reduced.

2. Query group creation with EM algorithm:

This algorithm is used to find the maximum likelihood estimates of parameters, and it performs two steps.

E-Step: Creates function for exceptions using current estimates.

M-Step: Computes the parameters maximizing the excepted log likelihood found on the E-Step.

Given a statistical model consisting of a set X of observed data, a set of missing values Z, and a vector of unknown parameters θ, along with a likelihood function $L(\theta;X,Z)=p(X,Z|\theta)$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data

$$L(\theta;X)=p(X|\theta)=\sum_Z p(X,Z|\theta)$$

However, this quantity is often intractable.

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation step (E step): conditional distribution of Z given X under the current estimate of the parameters θ(t):

$$Q(\theta|\theta(t))=E_{Z|X,\theta(t)}[\log L(\theta;X,Z)]$$

Maximization step (M step): Find the parameter that maximizes this quantity:

$$\theta(t+1)=\arg\max Q(\theta|\theta(t))$$

EM algorithms can be used for solving joint state and parameter estimation problems.

Example for dividing search histories in to Query groups shown in the below Fig1:

Time	Query
12.59.12	Saturn dealers
13.3.34	Saturn hybrid review
16.34.09	Bank of America
17.52.49	Caribben cruise
19.22.13	Gamestopdiscount
19.50.12	Tripadvisior
20.11.56	expedia

Time	Query
10.55.48	Saturn vue
10.52.24	Hybrid Saturn vue
11.00.42	snorkeling
11.12.78	Barbados hotel
12.17.23	Sprint slider phone
12.21.58	Toys us r wii
12.40.27	Best buy wii console
12.11.42	Financial statement
12.01.54	Wii gamestop

Group1	Group2	Group3
Saturn vue Hybrid Saturn vue Saturn dealers Saturn hybrid review	snorkeling Barbados hotel expedia	Sprint slider phone

Fig. 1. Search history of user activities with Query groups.

3. Query Relevance Using Search Logs

In this process, dividing the query relevance based on web search logs. In this our aim of relevance is capturing two properties of relevance queries, they are (i) queries that are frequently appear together as reformulations (ii) queries that have induced the users to click on similar set of pages. For that, we are introducing three search behaviour graphs that capture the aforementioned properties which we discuss in further.

3.1 Search Behaviour Graphs

In this there three types of graphs are there, they are Query Reformulation Graph (QRG), which represents the relationship between a pair of queries, which are reformulation of each other. The Query Click Graph (QCG), represents the relationship between two queries, which frequently lead to click on similar URLs. The Query Fusion Graph (QFG), which merges the above two graphs.

3.1.1 Query Reformulation Graph

In which, we are reformulating of pair of queries entered by the user. Here, we calculate pair of queries issued by the user, time based metric, simtime, which makes use of the internal between the time stamps of the queries in the user search history. Based on query logs we construct the query reformulation graph,QRG = (vq, ∑QR), and we calculate the Threshold value, Tr of two queries for each (qi,qj) with counterr (qi,qj)>= Tr We add a directed edge from qi to qj to Eqr. The Edge weight, Wr(qi,qj), is defined as the normalized count of the query transitions,

$$W_r(q_i;q_j) := \frac{count_r(q_i,q_j)}{\sum_{(q_i,q_k)\in\sum_{QR}} count_r(q_i,q_k)}$$

3.1.2 Query Click Graph

We construct the query click graph for the two queries may likely same, for which, we find relevance Query based upon the click. In which click graph (CG) has two sets of nodes corresponding to queries, VQ, and URLs, Vu extracted from the click logs. And there is an edge (qi, uk) Ɛ Ec, if Query qi was issued and uk was clicked by some users. In this we weight each edge(qi, uk) by the number of times qi was issued and uk was clicked, countr (qi, uk). In which we filters the query based on threshold Tr.. The weights of edge (qi, qj) in QCG, we (qi, qj), is defined as:

$$W_c(q_i,q_j) := \frac{\sum_{u_k} \min(count_c(q_i,u_k), count_c(q_c,u_k))}{\sum_{u_k} count_c(q_i,u_k)}$$

This captures the intuition that qj is more related to qiif more of qi's clicks fall on the URLs that are also clicked for qj

3.1.3 Query Fusion Graph

In which, we merge two graphs, those are QRG and QCG for better and effective relevance query, and store in a single graph, QFG = (vq,Eqf), and capture the weight of edge(qi,qj) in QRG, wf(qi,qj) and taken to be a linear sum of the edge weights, wr(qi,qj) in EQR and wc(qi,qj) in EQC, as follows:

$$w_f(q_i,q_j) = \alpha * w_r(q_i,q_j) + (1-\alpha) * w_c(q_i,q_j)$$

The relative contribution of the two weights is controlled by α, and we denote a query fusion graph constructed with a particular value of αas QFG(α).

Relevance(q)
Input:
1) the query fusion graph, $\mathcal{QFG}$
2) the jump vector, g
3) the damping factor, d
4) the total number of random walks, $numRWs$
5) the size of neighborhood, $maxHops$
6) the given query, q
Output: the fusion relevance vector for q, $\mathbf{rel}_q^F$
(0) Initialize $\mathbf{rel}_q^F = \mathbf{0}$
(1) $numWalks = 0$; $numVisits = 0$
(2) **while** $numWalks < numRWs$
(3) $numHops = 0$; $v = q$
(4) **while** $v \neq NULL \wedge numHops < maxHops$
(5) $numHops{+}{+}$
(6) $\mathbf{rel}_q^F(v){+}{+}$; $numVisits{+}{+}$
(7) $v = SelectNextNodeToVisit(v)$
(8) $numWalks{+}{+}$
(9) For each v, normalize $\mathbf{rel}_q^F(v) = \mathbf{rel}_q^F(v)/numVisits$

Fig2: Algorithm for calculating the query relevance by simulatingrandom walks over the query fusion graph.

3.2 Computing Query Relevance

The edges in QFG correspond to pair of relevant queries extracted from the query logs. It is not effective to use pairwise relevance values that directly expressed in QFG. Let us consider a vector rq, where each entry, rq(qj), is wf(q,qj) if there exists an edge from q to qj in QFG, and 0 otherwise. For capturing relevance of qj to q, is to use this rq(qj) value. It may work well in some cases, it will fail to capture relevance queries that are not directly connected in QFG.

SelectNextNodeToVisit(v)
Input:
1) the query fusion graph, $\mathcal{QFG}$
2) the jump vector, g
3) the damping factor, d
4) the current node, v
Output: the next node to visit, q_i
(0) **if** $random() < d$
(1) $V = \{q_i \mid (v,q_i) \in \mathcal{E}_{\mathcal{QF}}\}$
(2) pick a node $q_i \in V$ with probability $w_f(v,q_i)$
(3) **else**
(4) $V = \{q_i \mid g(q_i) > 0\}$
(5) pick a node $q_i \in V$ with probability $g(q_i)$
(6) **return** q_i

Fig 3 : Algorithm for selecting the next node to visit.

In which we are going to introduce the Markov chain for q, MCq , over the given graph, QRG & computing the stationary distribution of the chain. By this Markov chain approach we can find relevance queries easily. We refer this stationary distribution as the fusion relevance vector of q, relFq. The stationary probability distribution of MCq can be estimated using the matrix multiplication method, have the matrix corresponding to MCq is multiplied by itself iteratively until the resulting matrix reaches a fix point. We introduce the Monte Carlo Random walk simulation method to compute the stationary distribution whenever a new query comes in. The random walk simulation then proceeds as follow: we use the jump vector gq to pick the random walk starting point. At each node v, for a given damping factor d, the random walk either continues by following one of the outgoing edges of v with probability of d, or stops or restarts at one of the starting point in gq with a probability of (1-d). The selection of the next node to visit based on the outgoing edges of current node v in QFG & damping factor d is performed by the select next node to visit algorithm. Given query q and damping factor d, the marker chain for q, MCq, is defined as follows:

MCq(qi,qj) =d * wf (qi,qj) if qj ≠ q,

MCq(qi,qj) =d * wf (qi,qj)+(1- d) if qj = q

3.3 Incorporating Current Clicks

In addition to query reformulations, user activities also include clicks on the URLs following each query submission. The clicks of a user may further help us infer her search interests behind a query q and thus identify queries and query groups relevant to q more effectively. In this section, we explain how we can use the click information of the current user to expand the random walk process to improve our query relevance estimates. Note that the approach we introduce in this section is independent of modeling the query click information as QCG, to build QFG. Here, we use clicks of the current user to better understand her search intent behind the currently issued query, while clicks of massive users in the click logs are aggregated into QCG to capture the degree of relevance of query pairs through commonly clicked URLs.

We now describe how we use the clicked URLs by the current user together with the given query q to better capture her search intent. First, we identify the set of URLs, clk, which was clicked by the current user after issuing q. Then, we use clk and the click-through graph CG to expand the space of queries considered when we compute the fusion relevance vector of q. Unlike the jump vector that, reflects the given query q only, we now consider both q and clk together when we set a new jump vector.

Given q and clk, we employ a click jump vector, gclk that represents the queries in CG that have also induced clicks to the URLs within clk. Each entry in gclk,gclk(qi) corresponds to the relevance of query qi to the URLs in clk. Using CG,wedefine gclk as the proportion of the number of clicks to clkinduced by qi(qi∈vq\{q}) to the total number of clicks to clk induced by all the queries in vq\{q}

$$g_{clk}(q_i) := \frac{\sum_{u_k \in clk} count_c(q_i, u_k)}{\sum_{q_j \in v_Q, q_j \neq q} \sum_{u_k \in clk} count_c(q_j, u_k)}$$

Since the given query q is already captured in gq, we entry in gclkcorresponding to q to 0 (gclk(q)=0).

4. Conclusion:

By using EM algorithm, the user search history divide into few related query group in a dynamic and automated fashion. At that time, it is easy to the user to retrieve the required information depends upon the keyword entered by the user. And it will reduce the searching burden on the user. This EM algorithm provides the more efficient out come to the user.

REFERENCES

1. Heasoo Hwang, Hady W. Lauw, Lise Getoor, and Alexandros Ntoulas “Organizing User Search Histories”, ieee, may 2012. | 2. http://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm. | 3. J. Teevan, E. Adar, R. Jones, and M.A.S. Potts, “Information ReRetrieval:Repeat Queries in Yahoo’s Logs,” (SIGIR ’07). | 4. A. Broder, “A Taxonomy of Web Search,” SIGIR Forum, 2002. | 5. Spink, M. Park, B.J. Jansen, and J. Pedersen, “Multitasking during Web Search Sessions,” Information Processing and Management. | 6. R. Jones and K.L. Klinkner, “Beyond the Session Timeout:Automatic Hierarchical Segmentation of Search Topics in Query Logs,” Proc.