ISSN 2320 2629

Volume 4, No.1, January – February 2015

International Journal of Information Technology Infrastructure

Available Online at http://warse.org/pdfs/2015/ijiti01412015.pdf

Spatial Query Performance in cloud by using two from virtual machine

Ashraf Sabri Waheed Alameri

M.Sc (Computer Science) Mahatma Gandhi College Affiliated to Acharya Nagarjuna University Guntur, AP, India,ashfat2004@yahoo.com

ABSTRACT

Search Engines has always been the chosen mode of information retrieval (IR) systems. Users are no longer content with issuing simple navigational queries. A complex query such as travel arrangement has to be broken down into a number of co-dependent steps over a period of time. For instance, a user may first search on possible destinations, timeline, events, etc. After deciding when and where to go, the user may then search for the most suitable arrangements for air tickets, rental cars, lodging, meals, etc. Each step requires one or more queries, and each query results in one or more clicks on relevant pages. Keyword based search engines cannot support this kind of hierarchical queries. So I propose to use Random walk propagation methods that construct user profile based on his credentials from its user search history repositories. Combined with click points driven click graphs of user search behaviors the IR system can support complex queries for future requests at reduced times. Random walk propagation over the query fusion graph methods support complex search quests in IR systems at reduced times. For making the IR Systems effective and dynamic I also propose to use these search quests as auto complete features in similar query propagations. Biasing the ranking of search results can also be provided using any ranking algorithms (top-k algorithms).Supporting these methods yields dynamic performance in IR systems, by providing enriched user querying experience. A practical implementation of the proposed system validates our claim.

Key words: *query clustering, search engine, query reformulation, click graph, task identification*

1. INTRODUCTION

AS the size and richness of information on the Web grows, so does the variety and the complexity of tasks that users try to accomplish online. Users are no longer content with issuing simple navigational queries. Various studies on query logs (e.g., Yahoo's and AltaVista's reveal that only about 20% of queries are navigational. The rest are informational or transactional in nature. This is because users now pursue much broader informational and task-oriented goals such as arranging for future travel, managing their finances, or planning their purchase decisions[1].

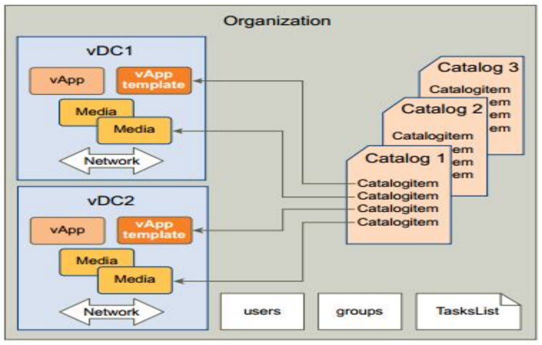

Figure 1: Data Retrieval over VM specification in recent application development

To improve user's search experience, most major commercial search engines provide query suggestions to help users formulate more effective queries. When a user submits a query, a list of terms that are semantically related to the submitted query is provided to help the user identify terms that he/she really wants, hence improving the retrieval effectiveness. Yahoo's "Also Try" and Google's "Searches related to" features provide related queries for Narrowing search, while Ask Jeeves suggests both more specific and more general queries to the user[2]. One important step towards enabling services and features that can help users during their complex search quests online is the capability to identify and group related queries together. Recently, some of the major search engines have introduced a new "Search History" feature, which allows users to track their online searches by recording their queries and clicks. This history includes a sequence of four queries displayed in reverse Chronological order together with their corresponding clicks. In addition to viewing their search history, users can manipulate it by manually editing and organizing related queries and clicks into groups, or by sharing them with their friends[3].

Once query groups have been identified, search engines can have a good representation of the search context behind the current query using queries and clicks in the corresponding query group. For example, if a search engine knows that a current query "financial statement" belongs to a {"bank of America", "financial statement"} query group, it can boost the rank of the page that provides information about how to get a Bank of America statement instead of the Wikipedia article on "financial statement", or the pages related to financial statements from other banks.

In this paper we motivate and propose a method to perform query grouping in a dynamic fashion. My goal is to ensure good performance while avoiding disruption of existing user-defined query groups. I investigate how signals from search logs such as query reformulations and clicks can be used together to determine the relevance among query groups. I study two potential ways of using clicks in order to enhance this process: by fusing the query reformulation graph and the query click graph into a single graph that I refer to as the *query fusion graph*, and by expanding the query set when computing relevance to also include other queries with similar clicked URLs, Figure 1.

2. RELATED WORK

Baeza-Yates et al proposed a query clustering method that groups similar queries according to their semantics. The method creates a vector representation Q or a query q, and the vector Q is composed of terms from the clicked documents of q. Cosine similarity is applied to the query vectors to discover similar queries[4]. More recently, Zhang and Nasraoui presented a method that discovers similar queries by analyzing users' sequential search behavior. The method assumes that consecutive queries submitted by a user are related to each other. The sequential search behavior is combined with a traditional contentbased similarity method to compensate for the high sparsity of real query log data.

Time	Query
10:51:45	Saturn Value
10:54:27	Hybrid Saturn value description
11:21:07	Will GameStop
12:22:22	Sprint Latest Model

Figure 2: User time results based on searching process

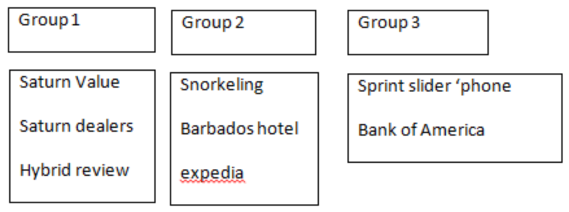

Figure 3: User processing results with semantic group results

My goal is to automatically organize a user's search history into query groups, each containing one or more related queries and their corresponding clicks. Each query group corresponds to an atomic information need that may require a small number of queries and clicks related to the same search goal. For example, in the case of navigational queries, a query group may involve as few as one query[5]. One major problem with the click through-based method is that the number of common clicks on URLs for different queries are limited. This is because different queries will likely retrieve very different result sets in very different ranking orders.

Dynami3c Query Grouping: One approach to the identification of query groups is to first treat every query in a user's history as a singleton query group, and then merge these singleton query groups in an iterative fashion (in a k-means or agglomerative way [8]). However, this is impractical in our scenario for two reasons. First, it may have the undesirable effect of changing a user's existing query groups, potentially undoing the user's own manual efforts in organizing her history. Second, it involves a high computational cost, since we would have to repeat a large number of query group similarity computations for every new query.

3. EXISTING APPROACH

Personalized Concept-Based Clustering: I now explain the essential idea of my personalized concept-based clustering algorithm with which ambiguous queries can be clustered into different query clusters. Personalized effect is achieved by manipulating the user concept preference profiles in the clustering process. In contrast to BB's agglomerative clustering algorithm, which represents the same queries submitted from different users by one query node, we need to consider the same queries submitted by different users separately to achieve personalization effect. In other words, if two given queries, whether they are identical or not, mean different things to two different users, they should not be merged together because they refer to two different sets of concepts for the two users. Therefore, we treat each individual query submitted by each user as an individual vertex in the bipartite graph by labeling each query with a user identifier[4].

4. PROPOSED APPROACH

A user queries a search engine Search Engine tries to construct user profile based on his ipaddress/login credentials from its user search history repositories. If the user already exists, the search engine checks from its user search history repositories up to a certain threshold whether the user already queried the same query previously If the user did, then search engine further retrieves click points from user search history repositories and reformulates query results by generating click graphs[6]. Click graphs contain useful information on user behavior when searching online. This step is called query fusion graph. Uses random walk propagation over the query fusion graph instead of time-based and keyword similarity based approaches. This entire process is called organizing user search histories into query groups. This approach helps users to pursue complex search quests online.

5. QUERY REKEVANCE USING SEARCH LOGS

I now develop the machinery to define the *query relevance* based on Web search logs. My measure of relevance is aimed at capturing two important properties of relevant queries[10], namely: (1) queries that frequently appear together as reformulations and (2) queries that have induced the users to click on similar sets of pages

```
Find the relevance
Input: QFG, factor, given query, q.
Output: Relevance vector for given query.
Step 1: Initially rel=0
Step 2: random walk propagation, number of visits.
Step 3: for each user processing results are
displayed based on numVisits
Step 4: above two steps are repeated to every user
processing in search process.
```

Figure 4: Algorithm for calculating the query relevance by simulating random walks over the query fusion graph

Search Behavior Graphs

I derive three types of graphs from the search logs of a commercial search engine. The *query reformulation graph*, QRG, represents the relationship between a pair of queries that are likely reformulations of each other. The *query click graph*, QCG, represents the relationship between two queries that frequently lead to clicks on similar URLs[9].

Query Reformulation Graph: One way to identify relevant queries is to consider *query reformulations* that are typically found within the query logs of a search engine. If two queries that are issued consecutively by many users occur frequently enough, they are likely to be reformulations of each other[7].

Query Click Graph: A different way to capture relevant queries from the search logs is to consider queries that are likely to induce users to click frequently on the same set of URLs. For example, although the queries “ipod” and “apple store” do not share any text or appear temporally close in a user’s search history, they are relevant because they are likely to have resulted in clicks about the iPod product.

Query Fusion Graph: The query reformulation graph, QRG, and the query click graph, QCG, capture two important properties of relevant queries respectively.

6. PERFORMANCE ANALYSIS

Experimental Setup:

I study the behavior and performance of my algorithms on partitioning a user’s query history into one or more groups of related queries. For example, for the sequence of queries “Caribbean cruise”; “bank of America”; “expedient”; “financial statement”, we would expect two output partitions: first, {“Caribbean cruise”, “expedia”} pertaining to travel-related queries, and, second, {“bank of America”, “financial statement”} pertaining to money-related queries.

Using Search Logs

My query grouping algorithm relies heavily on the use of search logs in two ways: first, to construct the query fusion graph used in computing query relevance, and, second, to expand the set of queries considered when computing query relevance. I start my experimental evaluation, by investigating how we can make the most out of the search logs. In my first experiment, I study *how I should combine* the query graphs coming from the query reformulations and the clicks within I query log[10].

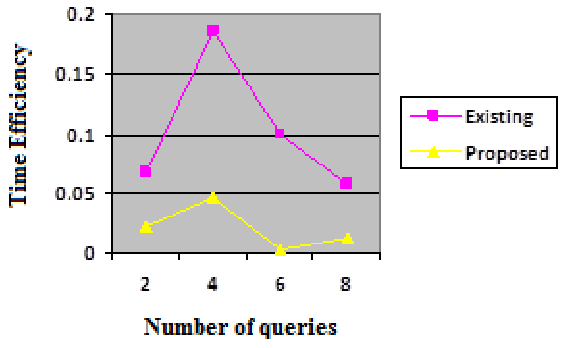

Figure 5: Varying query results in both existing and proposed approaches.

Above graph describes the horizontal axis represents _ (i.e., how much weight we give to the query edges coming from the query reformulation graph), while the vertical axis shows the performance of our algorithm in terms of the RandIndex metric.

7. CONCLUSION

The Query formulations based on click graphs contain useful information on user behavior when searching online. For this process I are using different informative techniques like page rank operations for analyzing the user histories. In this paper I propose to develop the efficient data extraction based on click graph results. I also find value in combining my method with keyword similarity-based methods, especially when there is insufficient usage information about the queries. As future work, I intend to investigate the usefulness of the knowledge gained from these query groups in various applications such as providing query suggestions and biasing the ranking of search results.

REFERENCES

1. http://www.dmoz.org/, 2008.
2. http://www.google.com/, 2008.
3. http://www.sigkdd.org/kdd2005/kddcup.html, 2008.
4. D. Beeferman and A. Berger, “Agglomerative Clustering of a Search Engine Query Log,” Proc. ACM SIGKDD, 2000.
5. R. Jones and K. L. Klinkner, “Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs,” in *CIKM*, 2008.
6. P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, “The query-flow graph: Model and applications,” in *CIKM*, 2008.
7. R. Baeza-Yates and A. Tiberi, “Extracting semantic relations from query logs,” in *KDD*, 2007.
8. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
9. P. Anick, “Using terminological feedback for web search refinement: A log-based study,” in *SIGIR*, 2003.
10. B. J. Jansen, A. Spink, C. Blakely, and S. Koshman, “Defining a session on Web search engines: Research articles,” *Journal of the American Society for Information Science and Technology*, vol. 58, no. 6, pp. 862–871, 2007.