Survey on Mining Web Graphs for Large Scale Meta Search Engine Results

[1]Bandi Krishna, [2]Dr. V.B.Narasimha
[1]Research Scholar, Dept. of CSE, Osmania University, Hyderabad, Telangana, India
[2]Asst.Prof, Dept. of CSE, Osmania University, Hyderabad, Telangana, India

Abstract
MSE is a search tool that sends user requests to several other search engines/databases, aggregates the results, merges/re-ranks them into a single list and displays them to user by using web graphs. MSEs enable users to enter search criteria once and access several search engines simultaneously. This also may save a lot of time to the user from having to use multiple search engines separately by initiating the search at a single point. Most of today's MSEs employ only a small number of general purpose search engines. Building large-scale MSEs using numerous specialized search engines is another area that deserves more attention. Challenges arising from building very large-scale MSEs include automatic generation and maintenance of high quality search engine representatives needed for efficient and effective search engine selection, and highly automated techniques to add search engines into MSEs and to adapt to changes of search engines. In this paper we are taking our study on how to merge the search results returned from multiple component search engines into a single ranked list; this is an important issue in MSE research. With the help of web graphs meta search engine can produce efficient results. web graphs are essential for developing effective and efficient large scale Meta Search engine results.

Keywords
Web Graphs, Meta search engine, Large Scale, Information retrieval, Algorithm

I. Introduction

To the web user, a Meta Search Engine (MSE) appears much like a regular search engine (SE). MSE, unlike an SE does not have an index. Instead, it dynamically queries multiple search engines; extracts, fuses and re-ranks results and presents to users.

Meta search engines create what is known as a virtual database. They do not compile a physical database or catalogue of the web. Instead, they take a user's request, pass it to several other heterogeneous databases and then compile the results in a homogeneous manner based on specific algorithm. MetaCrawler, Savvy Search and Mamma are some of the earliest Meta search engines.

Meta Search Engines (MSE) were proposed and built as data mining tools. Meta Search Engines (MSE) on internet has improved continually with application of new methodologies. Understanding and utilization of MSEs are valuable for computer scientists and researchers, for effective information retrieval.

Ex: MetaCrawler, Inquirus-2,
Vertical MSE, Nano-Spider, Cancer-Spider.

II. Related Work

No two Meta search engines are alike. Some search only the most popular search engines while others also search lesser-known engines, newsgroups, and other databases. They also differ in how the results are presented and the quantity of engines that are used. Some will list results according to search engine or database. Others return results according to relevance, often concealing which search engine returned which results. This benefits the user by eliminating duplicate hits and grouping the most relevant ones at the top of the list.

Search engines frequently have different ways they expect requests submitted. For example, some search engines allow the usage of the word "AND" while others require "+" and others require only a space to combine words. The better Meta search engines try to synthesize requests appropriately when submitting them

A Meta Search Engine acts as an agent for the participant search engine. It receives queries from users and redirects them to one or more of the participant search engines for processing. The various algorithms are used for search engine selection and result merging that provides relevant information according to the user.

A. Search Engine Selector

If the number of component search engines in a metasearch engine is very small, say less than 10, it might be reasonable to send each user query submitted to the metasearch engine to all the component search engines. In this case, the search engine selector is probably not needed. However, if the number of component search engines is large, as in the large scale Meta Search Engine scenario, then sending each query to all component search engines will be an inefficient strategy because most component search engines will be useless with respect to any particular query. For example, suppose a user wants to find 50 best matching results for his/her query from a metasearch engine with 1,000 component search engines. Since the 50 best results will be contained in no more than 50 component search engines, it is clear that at least 950 component search engines are useless for this particular query. Passing a query to useless search engines may cause serious problems for efficiency. Generally, sending a query to useless search engines will cause waste of resources to the metasearch engine server, each of the involved search engine servers and the Internet. Specifically, dispatching a query, including needed query reformatting, to a useless search engine and handling the returned results, including receiving the returned response pages, extracting the result records from these pages, and determining whether they should be included in the final merged result list and where they should be ranked in the merged result list if they are to be included, waste the resources of the metasearch engine server; receiving the query from the metasearch engine, evaluating the query, and returning the results back to the metasearch engine waste the resources of each search engine whose results end up useless; and finally transmitting a query from the metasearch engine to useless search engines and transmitting useless retrieved results from these search engines to the metasearch engine waste the network resources of the Internet. Therefore, it is important to send each user query to only potentially useful search engines for processing. The problem of identifying potentially useful component search

engines to invoke for a given query is the search engine selection problem, which is sometimes also referred to as database selection problem, server selection problem, or query routing problem. Obviously, for metasearch engines with more component search engines and/or more diverse component search engines, having an effective search engine selector is more important.

B. Search Engine Connectors

After a component search engine has been selected to participate in the processing of a user query, the search engine connector established a connection with the server of the search engine and passes the query to it. Different search engines usually have different connection parameters. As a result, a separate connector is created for each search engine. In general, the connector for a search engine S needs to know the HTTP (Hyper Text Transfer Protocol) connection parameters supported by S. There are three basic parameters, (a) the name and location of the search engine server, (b) the HTTP request method (usually it is either GET or POST) supported by S, and (c) the name of the string variable that is used to hold the actual query string. When implementing metasearch engines with a small number of component search engines, experienced developers can manually write the connector for each search engine. However, for large-scale metasearch engines, this can be very time-consuming and expensive. Thus, it is important to develop the capability of generating connectors automatically. An intelligent metasearch engine may modify a query it receives from a user before passing it to the search engine connector if such a modification can potentially improve the search effectiveness. For example, a query expansion technique may be used by the metasearch engine to add terms that are related to the original user query to improve the chance for retrieving more relevant documents.

C. Web Service

A Meta search engine is a search engine that collects results from other search engine. Web service offers such functionality and then presents a summary of that information as the results of a search. Most search engines available on the Web provide only a browser based interface; however, because Web services start to be successful, some of those search engines offer also an access to their information through Web services. Two types of search engines are observed, one that acts like a wrapper for the HTML pages returned by the search engine and other one is build upon the Web service offered by the search engine but this difference is visible only when looking at the internal processing of the service. It is difficult to distinguish them from the outside as they implement the same interface. Web service are built for, any process that can be integrated into external systems through valid XML documents over Internet protocols. This definition outlines the general idea of Web services. Web services can be seen as software components with an interface to communicate with other software components. They have a certain functionality that is available through a special kind of Remote Procedure Call. SOAP, the Simple Object Access Protocol [16] was developed to enable a communication between Web services. It was designed as a lightweight protocol for exchange of information in a decentralized, distributed environment. SOAP is an extensible, text-based framework for enabling communication between diverse parties that have no prior knowledge of each other. This is the requirement a transport protocol for Web services has to fulfill. SOAP species a mechanism to perform remote procedure calls and therefore removes the requirement that two systems must run on the same platform or be written in the same programming language.

D. Result Extractors

After a component search engine processes a query, the search engine will return one or more response pages. A typical response page contains multiple (usually 10) search result records, each of which corresponds to a retrieved Web page, and it typically contains the URL and the title of the page, a short summary (called snippet) of the page content, and some other pieces of information such as the page size. the upper portion of a response page from the Google search engine. Response pages are dynamically generated HTML documents, and they often also contain content unrelated to the user query such as advertisements (sponsored links) and information about the host Web site. A program (i.e., result extractor) is needed to extract the correct search result records from different component search engines can be merged into a single ranked list. This program is sometimes called an extraction wrapper. Since different search engines often format their results differently, a separate result extractor is usually needed for each component search engine. Although experienced programmers can write the extractors Manually, for large-scale metasearch engines, it is desirable to develop techniques that can generate the extractors automatically.

E. Result Merger

After the results from the selected component search engines are returned to the metasearch engine, the result merger combines the results into s single ranked list. The ranked list of search result records is then presented to the user, possible 10 records on each page at a time, just like more search engines do. Many factors may influence how result merging will be performed and what the outcome will look like. The information that could be utilized includes the local rank of a result record from a component search engine, the title and the snippet of a result record, the full document of each result, the publication time of each retrieved document, the potential relevance of the search engine with respect to the query from where a result is retrieved, and more. A good result merger should rank all returned results in descending order of their desirability. The existing architecture has many disadvantages in search engine selection, search engine connection and result extractors. We proposed a robust metasearch engine architecture using web services for heterogeneous and dynamic environment.

1. Click through Data Analysis

In the field of click through data analysis, the most common usage is for optimizing Web search results or rankings, Web search logs are utilized to effectively organize the clusters of search results by (1) Learning “interesting aspects” of a topic and (2) generating more meaningful cluster labels. Ranking function is learned from the implicit feedback extracted from search engine click through data to provide personalized search results for users. Besides ranking, click through data is also well studied in the query clustering problem. Query clustering is a process used to discover frequently asked questions or most popular topics on a search engine. This process is crucial for search engines based on question answering. Recently, click through data has been analyzed and applied to several interesting research topics such as Web query hierarchy building and extraction of class attributes. Relations between these queries in a hierarchy. A typical relationship can be learning from click through data is that “bmw” is a child of

"car." The method proposed in [13] can extract attributes such as "capital city" and "President" for the class "Country," or "cost," "manufacturer" and "side effects" for the class "Drug." The method initially relies on a small set of linguistically motivated extraction patterns applied to each entry from the query logs, then employs a series of Web-based precision-enhancement filters to refine and rank the candidate attributes. The proposed method consists of two stages: generating Search Results and Apply Web Graphs on particular results. So click through data analysis mainly focus on number of clicks for a link. Based on clicks only order of links present to the user.

2. Diffusion on Graphs

Diffusin on Graphs by Hao Ma, Irwin King and Michael Rung-Tsong Lyu, concentrated on a novel graph diffusion model based on heat diffusion. This model can be applied to both undirected graphs and directed graphs then analyze the computational complexity of the model.

- Heat Diffusion
- Diffusion on Undirected Graphs
- Diffusion on Directed Graphs
- Complexity Analysis

(i). Heat Diffusion

Heat diffusion is a physical phenomenon. In a medium, heat always flows from a position with high temperature to a position with low temperature. The query-click graph is a bipartite graph with queries on one side and clicked documents on the other hand diffusion-based approaches have been successfully applied in various domains such as classification and dimensionality reduction problems. They used heat diffusion to model the similarity information propagation on Web graphs. In Physics, the heat diffusion is always performed on a geometric manifold with initial conditions. However, it is very difficult to represent the Web as a regular geometry with a known dimension.

(ii). Diffusion on Undirected Graphs

The heat difference at a node between time $t+\Delta t$ and time t will be equal to the sum of the heat that it receives from all its neighbors.

(iii). Diffusion on Directed Graphs

In many situations, the Web graphs are directed, especially in online recommender systems or knowledge sharing sites. Every user in knowledge sharing sites typically has a trust list. The users in the trust list can influence this user deeply. These relationships are directed since user a is in the trust list of user b, but user b might not be in the trust list of user a. At the same time, the extent of trust relations is different since user ui may trust user uj with trust score 1 while trust user uk only with trust sco. Hence, there are different weights associated with the relations.

(iv). Complexity Analysis

The complexity shows that heat diffusion algorithm enjoys very good performance in scalability since it is linear with respect to the number of edges in the graph. However, since the size of Web information is very large, the graph built upon the Web information can become extremely large. Then, the complexity is also too high, and the algorithm becomes time consuming and inefficient to get a solution. To overcome this difficulty they first extract a sub graph starting from the heat sources. Given the heat sources, the sub graph is constructed by using depth-first search in the original graph. Then, the diffusion processes will be performed on this sub graph efficiently and effectively. Generally, it will not decrease the qualities of the heat diffusion processes since the nodes too far away from the heat sources are normally not related to the sources.

III. Proposed Work

MSEs employ only a small number of general purpose search engines. Building large-scale MSEs using numerous specialized search engines is another area that deserves more attention. Challenges arising from building very large-scale MSEs include automatic generation and maintenance of high quality search engine representatives needed for efficient and effective search engine selection, and highly automated techniques to add search engines into MSEs and to adapt to changes of search engines.

In this paper we are taking our study on how to merge the search results returned from multiple component search engines into a single ranked list; this is an important issue in MSE research. An effective and efficient result merging strategy is essential for developing effective large scale Meta Search systems.

Proposed framework for MSE is shown in Figure 1 that takes into account both ranking and clustering mechanisms for organizing and presenting web pages to the user. The whole process, from giving the user query, to getting the results are organized in the following modules.

A. User Interface

This module provides the way of interaction to the proposed framework. When a user gives a query to the MSE, then this query is further provided to the multiple SEs for searching the information on the web. The returned results of the SE are stored in the local database.

B. Relevancy Calculator (RC)

RC provides a relevancy score to each returned web page of a SE. This relevancy score is calculated for finding, to which extant the page is matching to the user query. Authors of the paper know that a relevant web page is more similar to other relevant WebPages than the irrelevant pages. While there are many ways to define the relevancy based "similarity". This may include VSM, Okapi, CDR etc.. All the three named algorithms are implemented for this paper. But authors chose the Okapi similarity measure ranking formula [17] for evaluation of the proposed approach because it can perform better for both single term query search and for multiple term query search. Okapi takes two web pages as an input and correspondingly provides a similarity score. Higher score indicates better matches.

C. Result Generator (RG)

RG is responsible for generating the required number of Links. Generation of results is based on the lower and upper relevancy score of the web pages provided by the RC module. It also decides the relevancy range of each cluster for which URLs to be assigned. The complete process of cluster generation is illustrated by the algorithm given in Figure 1. The generated results are purely based on the similarity rank of the retrieved web pages with web graphs.

D. Web Page Adjuster (WPA)

WPA is responsible for removal of duplicate web pages and

assignment of ranked web pages to the corresponding cluster. Organizing the ranked results in the clusters is not meant to replace the traditional way of representing the search results with the new one. Higher the relevancy rank of the web page then higher is the possibility for the web page to be placed on the top of the cluster results. The complete process of assignment of web pages to the clusters is illustrated by the algorithm given in fig. 1.

E. Web Graphs

After getting the results from different multiple search engines, we have to Apply web graphs on results which are extracted from search tool interface. Now we will get efficient results which are relevant to the user's request.

Algorithm

Efficiency Search Results through web graph (ESRW)

- **Input:** User query Q, downloaded web pages (WP), number of required clusters (NC).
- **Output:** Labeled results with ranked links of web pages.

// Start of algorithm (Figure-1)

Step 1. Get the downloaded WP of each SE separately.

Step 2. For each WP of each SE calculate the relevancy score (RS).

Step 3. If (pages are completed)

{

Exit ()

}

Else

{

For each page Pi of downloaded WP with RS

If (Pi is already visited)

{

Eliminate the web page and go for next iteration

}

Else

{

Include the web page and go for next iteration

}

}

Step 4. Return the ordered results
Step 5. Apply web graphs on results
Step 6. Re-rank the web graphs results based on in which page accurately retrieved

Step 7. Show the Results

Step 8. Stop

This algorithm will help to produce the efficient results.

IV. Conclusion

Finally we conclude that we are going to research on mining web graphs for large scale Meta search engine results that effectively merges and ranks the results based on graphs from multiple search engines in an effective manner for the user to view best results for their queries.

References

[1] Hao Ma, Irwin King,"Mining Web Graphs for Recommendations".
[2] Weiyi meng et a1.,"Building Efficient and Effective Meta Search Engines".
[3] Joeran Beel et a1.,"Academic Search Engine Optimization (ASEO): Optimizing Scholarly Literature for Google Scholar & Co".
[4] Larry Kerschberg et a1.,"Intelligent Web Search via Personalizable Meta-search Agents".
[5] Eric J. Glover et a1.,"Architecture of a Meta search Engine that Supports User Information Needs".
[6] Theodora Tsikrika et a1.,"Merging Techniques for Performing Data Fusion on the Web".
[7] Choon Hoong Ding et a1.,"Guided Google: A Meta Search Engine and its Implementation using the Google Distributed Web Services".
[8] Hossein Jadidoleslamy et a1.,"Search Result Merging and Ranking Strategies in Meta Search Engines: A Survey".
[9] Felipe Bravo-Marquez et a1.,"DOCODE-Lite: A Meta-Search Engine for Document Similarity Retrieval".
[10] Yiyao Lu et a1.,"Evaluation of Result Merging Strategies for Meta search Engines".
[11] Khaled Abd-El-Fatah Mohamed et a1.,"Merging Multiple Search Results Approach for Meta Search Engines".
[12] Naresh kumar et a1., "A Meta Search Engine Approach for Organizing Web Search Results using Ranking and Clustering".
[13] K.Srinivas et a1.,"Web Service Architecture for a Meta Search Engine".
[14] Andreas Meier et a1.,"Meta-Search Engine Analysis".
[15] King-Lup Liu et a1.,"AllInOneNews: Development and Evaluation of a Large-Scale News Meta search Engine".
[16] Zonghuan Wu et a1.,"Creating Customized Meta search Engines on Demand Using SE-LEGO".
[17] Adele E. Howe et a1.,"SAVVY SEARCH A Meta search Engine That Learns Which Search Engines to Query".
[18] John Mc Carthy et a1.,"The Page Rank Citation Ranking: Bringing Order to the Web".
[19] Steve Lawrence et a1.,"Inquirus, the NECI meta search engine".
[20] Thorsten Joachims et a1.,"Optimizing Search Engines using Click through Data".
[21] Zonghuan Wu1 et a1., "Towards a Highly-Scalable and Effective Met search Engine".

Bandi Krishna completed his MCA from KITS(wgl), KU Telangana, and Pursuing Ph.D from Osmania University, Telangana, India.

Dr.V.B.NARASIMHA Assistant Professor, Department of C.S.E University College of Engineering Osmania University Hyderabad, India.