INTRODUCTION TO METASEARCH ENGINES AND RESULT MERGING STRATEGIES: A SURVEY

Hossein Jadidoleslamy
Deptt. of Information Tech., Anzali International Branch, University of Guilan, Rasht, Iran

ABSTRACT

MetaSearch is utilizing multiple other search systems to perform simultaneous search. A MetaSearch Engine (MSE) is a search system that enables MetaSearch. To perform a MetaSearch, user query is sent to multiple search engines; once the search results returned, they are received by the MSE, then merged into a single ranked list and the ranked list is presented to the user. When a query is submitted to a MSE, decisions are made with respect to the underlying search engines to be used, what modifications will be made to the query and how to score the results. These decisions are typically made by considering only the user's keyword query, neglecting the larger information need. The cornerstone of their technology is their rank aggregation method. In other words, Result merging is a key component in a MSE. The effectiveness of a MSE is closely related to the result merging algorithm it employs. In this paper, we want to investigate a variety of result merging methods based on a wide range of available information about the retrieved results, from their local ranks, their titles and snippets, to the full documents of these results.

KEYWORDS: *Search, Web, MetaSearch, MetaSearch Engine, Merging, Ranking.*

I. INTRODUCTION

MetaSearch Engines (MSEs) are tools that help the user identify such relevant information. Search engines retrieve web pages that contain information relevant to a specific subject described with a set of keywords given by the user. MSEs work at a higher level. They retrieve web pages relevant to a set of keywords, exploiting other already existing search engines. The earliest MSE is the MetaCrawler system that became operational since June 1995 [5,16]. Over the last years, many MSEs have been developed and deployed on the web. Most of them are built on top of a small number of popular general-purpose search engines but there are also MSEs that are connected to more specialized search engines and some are connected to over one thousand search engines [1,10]. In this paper, we investigate different result merging algorithms; The rest of the paper is organized as: In Section 2 motivation, In Section 3 overview of MSE, Section 4 provides scientific principles of MSE, Section 5 discusses about why do we use MSE, Section 6 discusses architecture of MSE, Section 7 describes ranking aggregation methods, In Section the paper expresses key parameters to evaluating the ranking strategies, Section 9 gives conclusions and Section 10 present future works.

II. MOTIVATION

There are some primarily factors behind developing a MSE, are:

- The World Wide Web (WWW) is a huge unstructured corpus of information; MSE covers a larger portion of WWW;
- By MSE we can have the latest updated information;
- MSE increases the web coverage;
- Improved convenience for users;

- MSE provides fast and easy access to the desired search [5]; better retrieval effectiveness [2];
- MSE provides a broader overview of a topic [12];
- MSE has ability to search the invisible Web, thus increasing the precision, recall and quality of result;
- MSE makes the user task much easier by searching and ranking the results from multiple search engine;
- MSE provides a quick way to determine which search engines are retrieving the best match for user's information need [4].

III. OVERVIEW OF METASEARCH ENGINE

MSE search several engines at once; it does not crawl the web or maintain a database of web pages; instead, they act as a middle agent, passing the user's query simultaneously to other search engines or web directories or deep web, returning the results, collecting them, remove the duplicate links, merge and rank them into a single list and display it to the user [5,8]. Some samples of MSEs are Vivisimo, MetaCrawler, Dogpile, Mamma, and Turbo10.

a. Differences Between Search and MetaSearch

- MSE does not crawl the Web [2,4];
- MSE does not have a Database [4,10];
- MSE sends search queries to several search engines at once [2,5];
- MSE increased search coverage (but is limited by the engines they use with respect to the number and quality of results) and a consistent interface [6,12];
- MSE is an effective mechanism to reach deep web.

b. MetaSearch Engine Definition

- Dictionary meaning for Meta: more comprehensive, transcending;
- Accept the User query; Convert the query into the correct syntax for underlying search engines, launch the multiple queries, wait for the result; Analyze, eliminate duplicates and merge results; Deliver the post processed result to the users.
- A MSE allows you to search multiple search engines at once, returning more comprehensive and relevant results, fast [5,9];
- A search engine which does not gather its own information directly from web sites but rather passes the queries that it receives onto other search engines. It then compiles, summarizes and displays the found information;
- MSE is a hub of search engines/databases accessible by a common interface providing the user with results which may/may not be ranked independently of the original search engine/source ranking [6,10].

c. The Types of MetaSearch Engine

Different types of MetaSearch Engines (MSEs) are:

- MSEs which present results without aggregating them;
- Searches multiple search engines, aggregates the results obtained from them and returns a single list of results [1,3], often with duplicate removed;
- MSEs for serious deep digging.

d. MSE Issues

Some of most common issues in MSEs are as follows:

- Performing search engine/database selection [5,6];
- How to pass user queries to other search engines;
- How to identify correct search results returned from search engines; an optimal algorithm for implementing minimum cost bipartite matching;
- How to search results extraction, requiring a connection program and an extraction program (wrapper) for each component search engine [14];
- Expensive/time-consuming to produce/maintain wrapper;

- merging the results from different search sources;
- Different search engines produce result pages in different formats [6,8].

IV. SCIENTIFIC FUNDAMENTALS

a. Search Engine Selection

To enable search engine selection, some information that can represent the contents of the documents of each component search engine needs to be collected first. Such information for a search engine is called the representative of the search engine [5,17]. The representatives of all search engines used by the MSE are collected in advance and are stored with the MSE. During search engine selection for a given query, search engines are ranked based on how well their representatives match with the query. Different search engine selection techniques often use different types of representatives. A simple representative of a search engine may contain only a few selected key words or a short description. This type of representative is usually produced manually but it can also be automatically generated [5]. As this type of representatives provides only a general description of the contents of search engines, the accuracy of using such representatives for search engine selection is usually low. More elaborate representatives consist of detailed statistical information for each term in each search engine [5,9,17].

b. Automatic Search Engine Connection

In most cases, the HTML form tag of a MSE contains all information needed to make the connection to the search engines. The form tag of each search engine interface is usually pre-processed to extract the information needed for program connection and the extracted information is saved at the MSE [5,17]. After the MSE receives a query and a particular search engine, among possibly other search engines, is selected to evaluate this query, the query is assigned to the name of the query textbox of the search engine and sent to the server of the search engine using the HTTP request method. After the query is evaluated by the search engine, one or more result pages containing the search results are returned to the MSE for further processing.

c. Automatic Search Result Extraction

A result page returned by a search engine is a dynamically generated HTML page. In addition to the search result records (SRRs) for a query, a result page usually also contains some unwanted information/links [5]. It is important to correctly extract the SRRs on each result page. A typical SRR corresponds to a retrieved document and it usually contains the URL, title and a snippet of the document. Since different search engines produce result pages in different format, a separate wrapper program needs to be generated for each search engine [5,14]. Most of them analyze the source HTML files of the result pages as text strings or tag trees to find the repeating patterns of the SRRs.

d. Results Merging

Result merging is to combine the search results returned from multiple search engines into a single ranked list. There are many methods for merging/ranking search results; some of them are,

- Normalizing the scores returned from different search engines into values within a common range with the goal to make them more comparable [1,6,16]; the results from more useful search engines to be ranked higher.
- Using voting-based techniques.
- Downloading all returned documents from their local servers and compute their matching scores using a common similarity function employed by the MSE [1,6,17].
- Using techniques rely on features such as titles and snippets and so on [1].
- The same retrieved results from multiple search engines are more relevant to the query [1,5].

V. WHY ARE METASEARCH ENGINES USEFUL?

1. Why MetaSearch?

- Individual Search engines do not cover all the web;
- Individual Search Engines are prone to spamming [5];

- Difficulty in deciding and obtaining results with combined searches on different search engines [6];
- Data Fusion (multiple formats supported) and take less effort of user.

2. Why MetaSearch Engines?

- General search engines have difference in search syntax, frequency of updating, display results/search interface and incomplete database [5,16];
- MSE improves the search quality with comprehensive, efficient and one query queries all;
- MSE is good for quick search results overview with 1 or 2 keywords;
- MSE convenient to search different content sources from one page.

3. Key Applications of MetaSearch Engines

- Effective mechanism to search surface/deep web;
- MSE provides a common search interface over multiple search engines [5,10];
- MSE can support interesting special applications.

4. General Features of MetaSearch Engine

- Unifies the search interface and provides a consistent user interface; Standardizes the query structure [5];
- May make use of an independent ranking method for the results [6]; May have an independent ranking system for each search engine/database;
- MetaSearch is not a search for Meta data.

VI. METASEARCH ENGINE ARCHITECTURE

MSEs enable users to enter search criteria once and access several search engines simultaneously. This also may save (a lot of time) the user from having to use multiple search engines separately (by initiating the search at a single point). MSEs have virtual databases; they do not compile a physical database. Instead, they take a user's request, pass it to several heterogeneous databases and then compile the results in a homogeneous manner. No two MSEs are alike; they are different in component search engines, ranking/merging methods, search results presentation and etc.

a. Standard Architecture

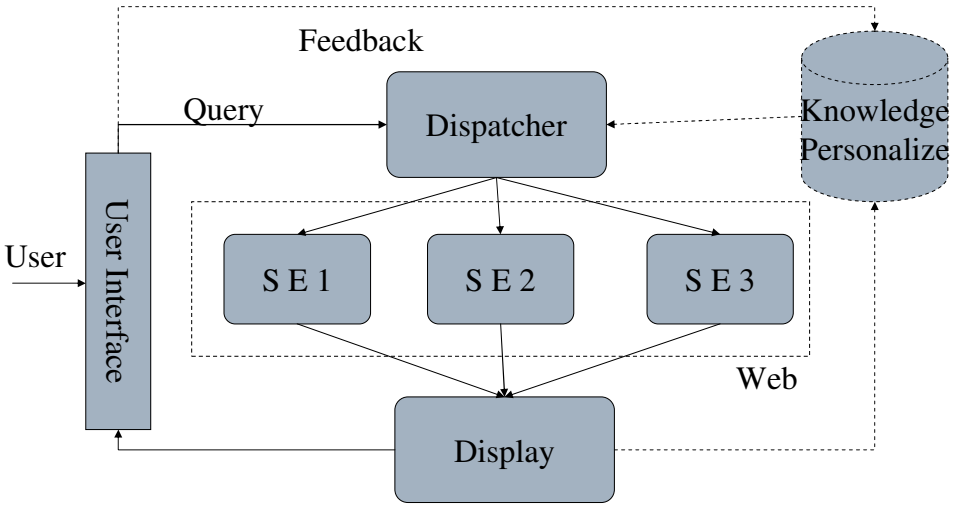

Figure1. Block diagram and components

- User Interface: similar search engine interfaces with options for types of search and search engines to use;
- Dispatcher: generates actual queries to the search engines by using the user query; may involve choosing/expanding search engines to use;

- Display: generates results page from the replies received; May involve ranking, parsing and clustering of the search results or just plain stitching;
- Personalization/Knowledge: may contain either or both. Personalization may involve weighting of search results/query/engine for each user.

b. The Architecture of a MSE with Concerns User Preferences

Current MSEs make several decisions on be-half of the user, but do not consider the user's complete information need. A MSE must decide which sources to query, how to modify the submitted query to best utilize the underlying search engines, and how to order the results. Some MSEs allow users to influence one of these decisions, but not all three [4,5].

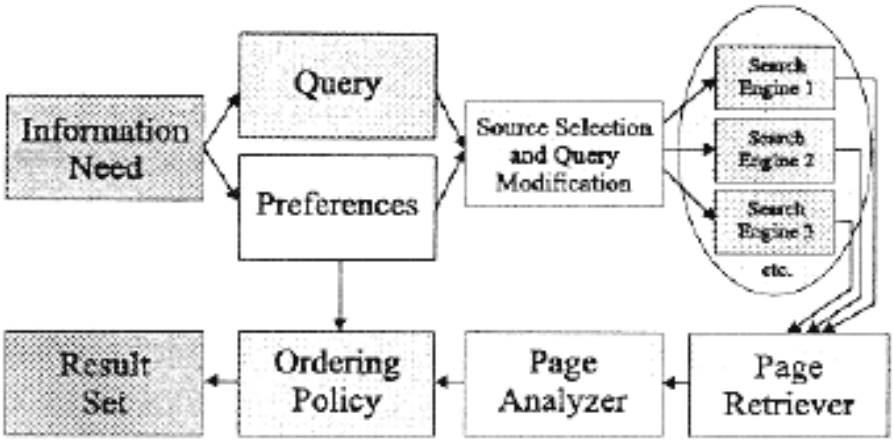

Figure2. The architecture of a MSE with user needs

User's information needs are not sufficiently represented by a keyword query alone [4,10]. This architecture has an explicit notion of user preferences. These preferences or a search strategy, are used to choose the appropriate search engines (source selection), query modifications and influence the order the results (result scoring). Allowing the user to control the search strategy can provide relevant results for several specific needs, with a single consistent interface [4]. The current user interface provides the user with a list of choices. The specification of preferences allows users with different needs, but the same query, to not only search different search engines (or the same search engines with different "modified" queries), but also have results ordered differently [4]. Sometimes Even though users have different information needs, they might type the same keyword query, and even search some of the same search engines. This architecture guarantees consistent scoring of results by downloading page contents and analyzing the pages on the server [1,4].

c. Helios Architecture

In this section we describe the architecture of Helios. The Web Interface allows users to submit their queries and select the desired search engines among those supported by the system. This information is interpreted by the Local Query Parser & Emitter that re-writes queries in the appropriate format for the chosen engines. The Engines Builder maintains all the settings necessary to communicate with the remote search engines. The HTTP Retrievers modules handle the network communications. Once search results are available, the Search Results Collector & Parser extracts the relevant information and returns it using XML. Users can adopt the standard Merger & Ranker module for search results or integrate their customized one [12].

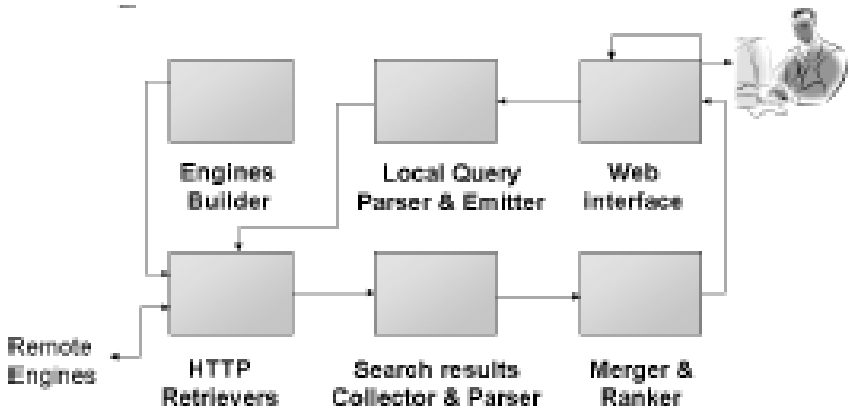

Figure3. The architecture of HELIOS MSE

d. Tadpole Architecture

In this architecture, when a user issues a search request, multiple threads are created in order to fetch the results from various search engines. Each of these threads is given a time limit to return the results, failing which a time out occurs and the thread is terminated [5,11].

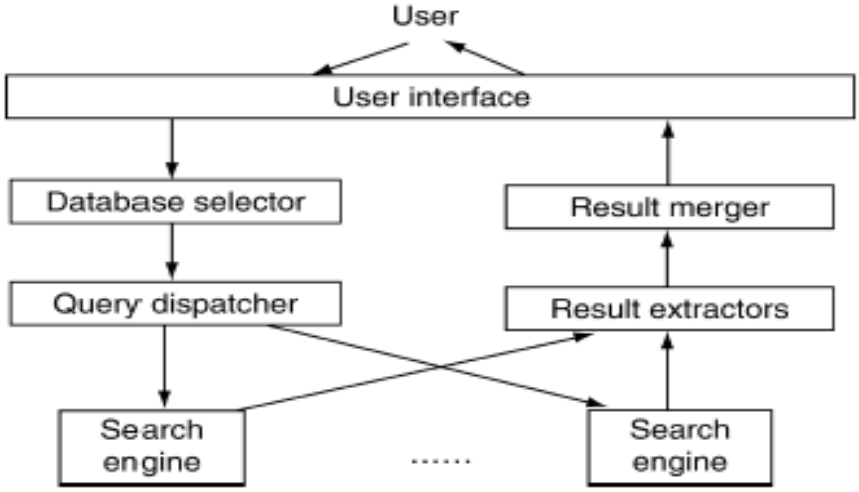

Figure4. Basic component architecture of a typical MSE

MSEs are web services that receive user queries and dispatch them to multiple crawl-based search engines; then collect returned results, reorder them and present the ranked result list to the user [11]. The ranking fusion algorithms that MSEs utilize are based on a variety of parameters, such as the ranking a result receives and the number of its appearances in the component engine's result lists [15]. Better results classification can be achieved by employing ranking fusion methods that take into consideration additional information about a web page. Another core step is to implicitly/explicitly collect some data concerning the user that submits the query. This will assist the engine to decide which results suit better to his informational needs [4,11,15].

VII. RESULTS MERGING AND RANKING STRATEGIES

There are many techniques for ranking retrieved search results from different search engines in MSEs; some important approaches are,

- Normalizing/ uniform the scores of search results[1];
- The reliability of each search engine;
- The document collection used by a search engine;

- Some ranking algorithms which completely ignore the scores assigned by the search engines to the retrieved web pages [1]: such as bayes-fuse and borda-fuse[7];
- Merging based on SRR contents such as title, snippet, local rank and different similarity functions[6];
- Considering the frequencies of query terms in each SRR, the order and the closeness of these terms;
- Downloading and analyzing full document.

We want to investigate result merging algorithms for MSEs. Most search engines present more informative search result records (SRRs) of retrieved results to the user; a typical SRR consists of the URL, title and snippet of the retrieved result [6,7].

1) Take the Best Rank

In this algorithm, we try to place a URL at the best rank it gets in any of the search engine rankings [13]. That is [17],

- MetaRank (x) = Min (Rank1(x), Rank2(x), …, Rankn(x));

Clashes are avoided by search engines popularity.

2) Borda's Positional Method

In this algorithm, MetaRank of an URL is obtained by computing the L1-Norm of the ranks in different search engines [8,17],

- MetaRank(x) =∑ (Rank1(x) p, Rank2(x) p, …, Rankn(x) p) 1/p;

Clashes are avoided by search engine popularity.

3) Weighted Borda-Fuse

In this algorithm, search engines are not treated equally, but their votes are considered with weights depending on the reliability of each search engine. These weights are set by the users in their profiles. Thus, the votes that the i result of the j search engine receive are [9,17],

- V (ri,j) = wj * (maxk (rk)-i+1);

Where wj is the weight of the j search engine and rk is the numbers of results rendered by search engine k. Retrieved pages that appear in more than one search engines receive the sum of their votes.

4) The Original KE Algorithm

KE Algorithm on its original form is a score-based method [1]. It exploits the ranking that a result receives by the component engines and the number of its appearances in the component engines' lists. All component engines are treated equally, as all of them are considered to be reliable. Each returned ranked item is assigned a score based on the following formula [10],

- Wke = ∑mi=1(r (i)) / ((n) m * (k/10 + 1) n);

Where ∑mi=1(r(i)) is the sum of all rankings that the item has taken, n is the number of search engine top-k lists the item is listed in, m is the total number of search engines exploited and k is the total number of ranked items that the KE Algorithm uses from each search engine. Therefore, it is clear that the less weight a result scores the better ranking it receives.

5) Fetch Retrieved Documents

A straightforward way to perform result merging is to fetch the retrieved documents to the MSE and compute their similarities with the query using a global similarity function. The main problem of this approach is that the user has to wait a long time before the results can be fully displayed. Therefore, most result merging techniques utilize the information associated with the search results as returned by component search engines to perform merging. The difficulty lies in the heterogeneities among the component search engines.

6) Borda Count

Borda Count is a voting-based data fusion method [15]. The returned results are considered as the candidates and each component search engine is a voter. For each voter, the top ranked candidate is assigned n points (n candidates), the second top ranked candidate is given n–1 points, and so on. For candidates that are not ranked by a voter (i.e., they are not retrieved by the corresponding search

engine), the remaining points of the voter will be divided evenly among them. The candidates are then ranked on their received total points in descending order [13,15,17].

7) D-WISE Method

In D-WISE, the local rank of a document (r_i) returned from search engine j is converted to a ranking score (rs_{ij}); the formula is [6],

- rsij = 1 – (ri - 1) * Smin / (m * Sj) ;

Where Sj is the usefulness score of the search engine j, Smin is the smallest search engine score among all component search engines selected for this query and m is the number of documents desired across all search engines. This function generates a smaller difference between the ranking scores of two consecutively ranked results retrieved from a search engine with a higher search engine score. This has the effect of ranking more results from higher quality search engines higher. One problem of this method is that the highest ranked documents returned from all the local systems will have the same ranking score 1.

8) Merging Based on Combination Documents Records (SRRs)

Among all the proposed merging methods, the most effective one is based on the combination of the evidences of document such as title, snippet, and the search engine usefulness. In these methods [1,2]: for each document, computing the similarity between the query and its title and its snippet; aggregating linearly the two as this document's estimated global similarity. For each query term, computing its weight in every component search engine based on the Okapi probabilistic model [6]. The search engine score is the sum of all the query term weights of this search engine. Finally, the estimated global similarity of each result is adjusted by multiplying the relative deviation of its source search engine's score to the mean of all the search engine scores. It is very possible that for a given query, the same document is returned from multiple component search engines. In this case, their (normalized) ranking scores need to be combined [1]. A number of linear combination fusion functions have been proposed to solve this problem include min, max, sum, average and etc [15].

9) Use Top Document to Compute Search Engine Score (TopD)

Assume Sj denote the score of search engine j with respect to q. This algorithm uses the similarity between q and the top ranked document returned from search engine j (denoted dij) [6,7]. Fetching the top ranked document from its local server have some delay, but that this delay is tolerable, since only one document is fetched from each used search engine. The similarity function using the Cosine function and Okapi function. The formula is [6],

- ∑TEq W * (((K1 + 1) * tf) / (K + tf)) * (((K3 + 1) * qtf) / (K3 + qtf)) ;
- With W = Log ((N-n+0.5) /(n+0.5)) and K = K1 * ((1-b)+b*(dl/avgdl)) ;

Where tf is the frequency of the query term T within the processed document, qtf is the frequency of T within the query, N is the number of documents in the collection, n is the number of documents containing T, dl is the length of the document, and avgdl is the average length of all the documents in the collection. K1, k3 and b are the constants with values 1.2, 1,000 and 0.75, respectively [6]. N, n, and avgdl are unknown, we can use some approximations to estimate them. The ranking scores of the top ranked results from all used search engines will be 1[1,6]. We remedy this problem by computing an adjusted ranking score arsij by multiplying the ranking score computed by above formula, namely rsij, by Sj [6], arsij = ∑ (rsij * Sj); If a document is retrieved from multiple search engines, we compute its final ranking score by summing up all the adjusted ranking scores.

10) Use Top Search Result Records (SRRs) to Compute Search Engine Score (TopSRR)

In this method, when a query q is submitted to a search engine j, the search engine returns the SRRs of a certain number of top ranked documents on a dynamically generated result page. In the TopSRR algorithm, the SRRs of the top n returned results from each search engine, instead of the top ranked document, are used to estimate its search engine score [6]. Intuitively, this is reasonable as a more useful search engine for a given query is more likely to retrieve better results which are usually reflected in the SRRs of these results. Specifically, all the titles of the top n SRRs from search engine j are merged together to form a title vector TVj, and all the snippets are also merged into a snippet vector SVj. The similarities between query q and TVj, and between q and SVj are computed separately and then aggregated into the score of search engine j [6],

- Sj = C1 * Similarity (q, TVj) + (1 – C1) * Similarity (q, SVj);

Too, both the Cosine function and Okapi function are used [6,7].

11) Compute Simple Similarities between SRRs and Query (SRRsim)

We can rank SRRs returned from different search engines; because each SRR can be considered as the representative of the corresponding full document. In the SRRsim algorithm, the similarity between a SRR (R) and a query q is defined as a weighted sum of the similarity between the title (T) of R and q and the similarity between the snippet (S) of R and q [6,7],

- Sim(R , q) = C2 * Similarity (q, T) + (1 – C2) * Similarity (q , S) ;

Where, C2 is constant (C2 = 0.5). Again both the Cosine function and the Okapi function are used. If a document is retrieved from multiple search engines with different SRRs (different search engines usually employ different ways to generate SRRs), then the similarity between the query and each such SRR will be computed and the largest one will be used as the final similarity for merging.

12) Rank SRRs Using More Features (SRRRank)

The similarity function used in the SRRsim algorithm may not be sufficiently powerful in reflecting the true matches of the SRRs with respect to a given query [6]. For example, these functions do not take proximity information such as how close the query terms occur in the title and snippet of a SRR into consideration, nor does it consider the order of appearances of the query terms in the title and snippet. Somtimes, the order and proximity information have a significant impact on the match of phrases. This algorithm defines five features with respect to the query terms; that are [6,7],

- NDT: The number of distinct query terms appearing in title and snippet;
- TNT: total number occurrences of the query terms in the title and snippet;
- TLoc: The locations of the occurred query terms;
- ADJ: whether the occurred query terms appear in the same order as they are in the query and whether they occur adjacently;
- WS: the window size containing distinct occurred query terms.

For each SRR of the returned result, the above pieces of information are collected. The SRRRank algorithm works as [6]:

- All SRRs are grouped based on NDT. The groups having more distinct terms are ranked higher;
- Within each group, the SRRs are further put into three subgroups based on TLoc. The subgroup with these terms in the title ranks highest, the subgroup with the distinct terms in the snippet and the subgroup with the terms scattered in both title and snippet;
- Finally, within each subgroup, the SRRs that have more occurrences of query terms (TNT) appearing in the title and the snippet are ranked higher. If two SRRs have the same number of occurrences of query terms, first the one with distinct query terms appearing in the same order and adjacently (ADJ) as they are in the query is ranked higher, and then, the one with smaller window size is ranked higher.

If there is any tie, it is broken by the local ranks. The result with the higher local rank will have a higher global rank in the merged list. If a result is retrieved from multiple search engines, we only keep the one with the highest global rank [3,6].

13) Compute Similarities between SRRs and Query Using More Features (SRRSimMF)

This algorithm is similar to SRRRank except that it quantifies the matches based on each feature identified in SRRRank so that the matching scores based on different features can be aggregated into a numeric value [1,3]. Consider a given field of a SRR, say title (the same methods apply to snippet). For the number of distinct query terms (NDT), its matching score is the ratio of NDT over the total number of distinct terms in the query (QLEN), denoted SNDT=NDT/QLEN. For the total number of query terms (TNT), its matching score is the ratio of TNT over the length of title, denoted STNT=TDT/TITLEN. For the query terms order and adjacency information (ADJ), the matching score SADJ is set to 1 if the distinct query terms appear in the same order and adjacently in the title; otherwise the value is 0. The window size (WS) of the distinct query terms in the processed title is converted into score SWS= (TITLEN–WS)/TITLEN. All the matching scores of these features are

aggregated into a single value, which is the similarity between the processed title T and q, using this formula [6],

- Sim(T , q) = SNDT + (1/QLEN) * (W1 * SADJ + W2 * SWS + W3 * STNT) ;

For each SRR, the final similarity is,

- Similarity = (TNDT/QLEN) * (C3 * Sim(T , q) + (1 – C3) * Sim (S , q)) ;

Where TNDT is the total number of distinct query terms appeared in title and snippet [6,7].

VIII. EVALUATION KEY PARAMETERS FOR RANKING STRATEGIES

Some parameters for ranking methods are algorithmic complexity (time complexity), rank aggregation time, overlap across search engines (relative search engine performance) and performance of the various rank aggregation methods include precision with respect to number of results returned and precision vs. recall.

IX. CONCLUSION

In this paper, we have presented an overview and some ranking strategies on MSEs. An effective and efficient result merging strategy is essential for developing effective MetaSearch systems. We investigated merging algorithms that utilize a wide range of information available for merging, from local ranks by component search engines, search engine scores, titles and snippets of search result records to the full documents. We discuss methods for improving answer relevance in MSEs; propose several strategies for combining the ranked results returned from multiple search engines. Our study has several results; that are,

- A simple and efficient merging method can help a MSE significantly outperform the best single search engine in effectiveness [2];
- Merging based on the titles and snippets of returned search result records can be more effective than using the full documents. This implies that a MSE can achieve better performance than a centralized retrieval system that contains all the documents from the component search engines;
- The computational complexity of ranking algorithms used and performance of the MSE are conflicting parameters;
- MSEs are useful, because,
  - Integration of search results provided by different engines; Comparison of rank positions;
  - Advanced search features on top of commodity engines;
  - A MSE can be used for retrieving, parsing, merging and reporting results provided by other search engines.

X. FUTURE WORKS

Component search engines employed by a MSE may change their connection parameters and result display format anytime. These changes can make the affected search engines unusable in the MSE. How to monitor the changes of search engines and make the corresponding changes in the MSE automatically. Most of today's MSEs employ only a small number of general purpose search engines. Building large-scale MSEs that using numerous specialized search engines is another area problem. 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 adapt to changes of search engines.

REFERENCES

[1] Renda M. E. and Straccia U.; Web metasearch: Rank vs. score based rank aggregation methods; 2003.

[2] Meng W., Yu C. and Liu K.; Building efficient and effective metasearch engines; In ACM Computing Surveys; 2002.

[3] Fagin R., Kumar R., Mahdian M., Sivakumar D. and Vee E.; Comparing and aggregating rankings with ties; In PODS; 2004.

[4] Glover J. E., Lawrence S., Birmingham P. W. and Giles C. L.; Architecture of a Metasearch Engine that Supports User Information Needs; NEC Research Institute, Artificial Intelligence Laboratory, University of Michigan; In ACM; 1999.

[5] MENG W.; Metasearch Engines; Department of Computer Science, State University of New York at Binghamton; Binghamton; 2008.

[6] Lu Y., Meng W., Shu L., Yu C. and Liu K.; Evaluation of result merging strategies for metasearch engines; 6th International Conference on Web Information Systems Engineering (WISE Conference); New York; 2005.

[7] Dwork C., Kumar R., Naor M. and Sivakumar D.; Rank aggregation methods for the Web; Proceedings of ACM Conference on World Wide Web (WWW); 2001.

[8] Fagin R., Kumar R., Mahdian M., Sivakumar D. and Vee E.; Comparing partial rankings; Proceedings of ACM Symposium on Principles of Database Systems (PODS); 2004.

[9] Fagin R., Kumar R. and Sivakumar D.; Comparing top k lists; SIAM Journal on Discrete Mathematics; 2003.

[10] Souldatos S., Dalamagas T. and Sellis T.; Captain Nemo: A Metasearch Engine with Personalized Hierarchical Search Space; School of Electrical and Computer Engineering; National Technical University of Athens; November, 2005.

[11] Mahabhashyam S. M. and Singitham P.; Tadpole: A Meta search engine Evaluation of Meta Search ranking strategies; University of Stanford; 2004.

[12] Gulli A., University of Pisa, Informatica; Signorini A., University of Iowa, Computer Science; Building an Open Source Meta Search Engine; May, 2005.

[13] Aslam J. and Montague M.; Models for Metasearch; In Proceedings of the ACM SIGIR Conference; New Orleans; 2001.

[14] Zhao H., Meng W., Wu Z., Raghavan V. and Yu C.; Fully automatic wrapper generation for search engines; World Wide Web Conference; Chiba, Japan; 2005.

[15] Akritidis L., Katsaros D. and Bozanis P.; Effective Ranking Fusion Methods for Personalized Metasearch Engines; Department of Computer and Communication Engineering, University of Thessaly; Panhellenic Conference on Informatics (IEEE); 2008.

[16] Manning C. D., Raghavan P. and Schutze H.; Introduction to Information Retrieval; Cambridge University Press; 2008.

[17] Dorn J. and Naz T.; Structuring Meta-search Research by Design Patterns; Institute of Information Systems, Technical University Vienna, Austria; International Computer Science and Technology Conference; San Diego; April, 2008.

Author Biography

H. Jadidoleslamy is a Master of Science student at the Guilan University in Iran. He received his Engineering Degree in Information Technology (IT) engineering from the University of Sistan and Balouchestan (USB), Iran, in September 2009. He will receive his Master of Science degree from the University of Guilan, Rasht, Iran, in March 2011. His research interests include Computer Networks (especially Wireless Sensor Network), Information Security, and E-Commerce. He may be reached at tanha.hossein@gmail.com.