References

[1] E. Selberg, and O. Etzioni. *Multi-Service Search and Comparison Using the MetaCrawler.* 4th International World Wide Web Conference, December 1995.

[2] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* International Conferences on Very Large Data Bases, 1995.

[3] L. Gravano, and H. Garcia-Molina. *Merging Ranks from Heterogeneous Internet Sources.* International Conferences on Very Large Data Bases, 1997.

[4] W. Meng, K.L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. *Determining Text Databases to Search in the Internet.* International Conferences on Very Large Data Bases, New York City, 1998.

[5] C. Yu, W. Meng, K. Liu, W. Wu, and N. Rishe. *Efficient and Effective Metasearch for a Large Number of Text Databases.* 8th ACM International Conference on Information and Knowledge Management (CIKM'99), November 1999.

Cross Reference:

Distributed Information Retrieval *see* Metasearch Engine.
Text Resource Discovery *see* Metasearch Engine.
Collection Fusion *see* Metasearch Engine.

sented using a rough description, such as a few words or a few paragraphs. Usually, the database representatives are formed manually. ALIWEB and Search Broker are example metasearch engines using this approach. This type of approach is very scalable to the size of individual databases and to the number of these databases. But its precision is usually low.

Detailed representative approach: Such an approach represents the content of a database using rather detailed statistical information. Typically, the representative of a database contains some information about each term in the database such as the *document frequency* of the term and the average weight of the term among all documents that have the term. Detailed statistics allow more accurate estimation of database usefulness with respect to any user query. Scalability of such approaches is an important issue due to the amount of information that need to be stored for each database. Several metasearch engines, including gGlOSS [2], CORI Net and D-WISE, employ various detailed representative approaches for database selection.

Learning-based approach: In such an approach, the knowledge about which databases are likely to return useful documents to what type of queries is learned either through training queries and/or users' reactions to the returned documents of their previous queries. Such knowledge is then used to determine the usefulness of databases for future queries. SavvySearch and ProFusion are example metasearch engines that employ learning-based database selection method.

Quantitative Approaches: These approaches are similar to the qualitative approaches using detailed database representatives. The main difference is that in a quantitative approach, the database selection algorithm estimates a quantity that can more explicitly and directly reflect the usefulness of a database with respect to a query than a ranking score. One such quantity is the number of documents in a database whose global similarity with a query is higher than a given threshold [4]. Another quantity is the global similarity of the most similar document with a query in a database [5]. The latter quantity can be used to rank databases optimally [5].

Collection Fusion

Various heterogeneities exist among individual search engines as they are likely to be developed independently. These heterogeneities include different term weighting schemes and different similarities functions. As a result, document selection and result merging should not be based on local document similarities alone as these similarities are often not comparable. Indeed, many search engines do not even provide local document ranking scores or similarities. Furthermore, search engines are often autonomous and would not reveal how local documents are scored or ranked.

To overcome the above problems and to enable collection fusion to be performed in a meaningful way, several measures can be taken. First, a global standard such as a global similarity function can be defined to reflect the user's intention as perceived by the metasearch engine. Second, knowledge discovery techniques could be used to estimate approximately how local search engines score or rank their documents against user queries. One way to accomplish this is to submit carefully designed probing queries to local search engines and analyze the results (the set of documents that are returned as well as the order in which they are displayed for each probing query). Such knowledge can provide insight on the relationship between the global similarities and the local similarities of documents from local search engines. The above insight or relationship can help the metasearch engine perform collection fusion in a number of ways. For example, a good local similarity threshold for a local search engine may be determined. By retrieving from this search engine only those documents whose local similarities are higher than this threshold, we can maximize the return of desired documents while minimize the return of useless documents. Finally, the ranking scores or ranks of documents provided by independent search engines can be combined to produce a global score or rank. Intuitively, that a document is ranked high by several search engines with different ranking techniques increases the likelihood that the document is relevant to the user and therefore deserved to be ranked high globally.

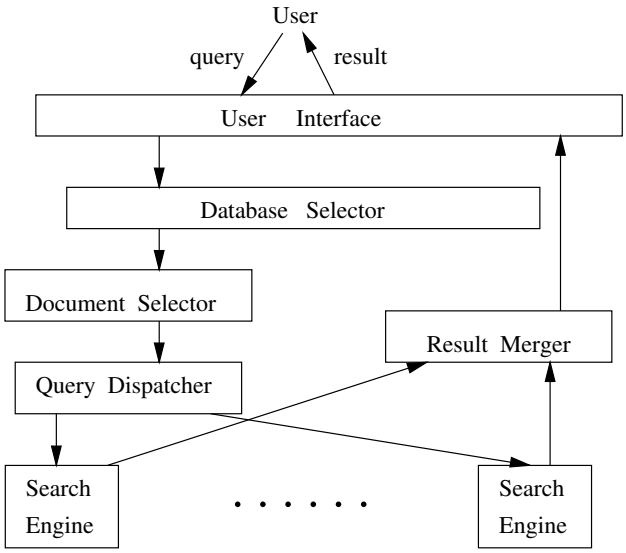

Figure 1: Metasearch Software Component Architecture

from the global similarity function which reflects the user's intention as perceived by the metasearch engine. The metasearch engine can have a *document selection module* for choosing the documents to retrieve from a search engine and for deciding which of the retrieved documents are to be returned to the metasearch engine. Document selection and result merging together determine for each query which documents from which database should be displayed to the user and are known as *collection fusion*. A reference software component architecture of a metasearch engine is shown in Figure 1.

Database Selection

Each search engine underneath a metasearch engine has a *text database* that is defined by the set of documents that can be searched by the search engine. Database selection is to identify for each given query the text databases that are likely to contain useful documents to the query. Database selection is sometimes also known as *text resource discovery*. As mentioned above, database selection is very important to the efficiency of a metasearch engine when the number of underlying search engines is large. The basic idea of performing database selection is as follows. First, some characteristic information about each database is obtained to indicate the approximate content of the database. The characteristic information is also known as the *database representative*. Next, when a user query is received, it is compared with the representative of each database to determine which database is likely to contain useful documents to the query.
Many database selection approaches exist. They range from choosing all underlying databases to selecting only useful databases based on some *usefulness* measure. These approaches can be classified into the following categories.

Naive approach: This approach does not measure the usefulness of underlying databases. The query dispatcher simply sends each user query to all underlying search engines indiscriminately. MetaCrawler is a metasearch engine employing this approach [1]. Naive approach is suitable when the metasearch engine has a very small number of underlying search engines.

Qualitative approaches: For a given user query, such an approach assigns a ranking score to each underlying database using the database representatives. Often, the ranking score reflects the quality of a database with respect to a query, but it may be difficult for ordinary users to interpret the meaning of such a score. Most database selection methods fall into this category and these approaches can be further classified into the following subcategories.

Rough representative approach: In such an approach, the content of a local database is repre-

METASEARCH ENGINE *

Clement Yu[1] and Weiyi Meng[2]
[1] Dept. of EECS, University of Illinois at Chicago, Chicago, IL 60607
[2] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902

Nowadays, millions of people all over the world use the World Wide Web to acquire needed information everyday. A very popular way to find desired information from the Web is to use a *search engine*, such as Altavista and Lycos. Upon receiving a user query which is usually in the form of a set of keywords or a natural language sentence, a search engine finds documents which match all or a significant portion of the keywords in the user query. Usually, a search engine can only evaluate queries against the collection of documents that it has preprocessed/indexed in advance.

The size of the Web has been increasing very rapidly. It was estimated that by the end of 1999, there were over one trillion publicly indexable Web pages. One consequence of this phenomenon is that no single search engine can keep up with the Web expansion. The relative coverage of the Web by individual search engines has been declining steadily. One effective way to increase the coverage of the Web is to combine the coverages of individual search engines. Incidentally, desired documents of a user are often scattered in a number of search engines. It is rather inconvenient for such a user to find all search engines that may contain useful documents to his/her query, to submit the query to all these search engines and to finally search through the results returned by these search engines to find desired documents. The task of combining the coverages of individual search engines and the task of allowing a user to submit a single query to invoke multiple search engines can both be achieved by the use of a *metasearch engine*. A metasearch engine is a layer of software (and possibly involving some hardware) which lies between the user and the different underlying search engines. Its function is to make a user feel like he/she is interacting with a single search engine but this software layer is capable of searching information from the union of the underlying search engines. In this article, we sketch techniques which are used to build metasearch engines.

Architecture

A metasearch engine may have a number of modules. Its *user interface module* accepts the user's query which will be forwarded by the *query dispatcher module* to the various search engines. When necessary, the query dispatcher will format the user query to satisfy the format requirement of each underlying search engine. When the search engines return the sets of retrieved documents to the metasearch engine, these sets are merged by the *result merger module* into a single ranked list of documents. Certain number of the top documents from this list are displayed to the user. When the number of search engines underlying a metasearch engine is large, forwarding each user query to every search engine is very inefficient. Often for a given query, many search engines do not contain any desired documents. Sending the query to these search engines not only incurs unnecessary network traffic but also causes these search engines to waste their local computing resources. Furthermore, these search engines may still return some (eventually useless) results to the result merger module of the metasearch engine, incurring additional communication cost and the effort to merge the results. To overcome these problems, a *database selection module* can be included in the metasearch engine and the function of this module is to identify for each user query the search engines that are likely to return useful documents to the user. With the help of this module, the query dispatcher forwards each user query to only those search engines that are likely to return useful results. Even when a search engine contains some useful documents, it may have to retrieve substantially more documents in order to guarantee that the desired documents are included in the result. This is due to the fact that each search engine may utilize a similarity function to retrieve documents and this similarity function is different

*Work supported in part by the following NSF grants: CCR-9816633, CCR-9803974, IIS-9902872, IIS-9902792.