Search Engine Selection Approach in Metasearch Using Past Queries

SHEO DAS¹ and KULDEEP SINGH RAGHUWANSHI²

¹M.Tech Student, AIET Jaipur, India.
²Professor CSE Department, AIET Jaipur, India.

(Received: March 23, 2014; Accepted: April 05, 2014)

ABSTRACT

A meta-search engine is a search engine that utilizes multiple search engines, when a meta search engine receives a query from a user, it invokes the underlying search engines to retrieve useful information for the user. Most of the data on the web is in the form of text or image. A good database selection algorithm should identify potentially useful databases accurately. Many approaches have been proposed to tackle the database selection problem. These approaches differ on the database representatives they use to indicate the contents of each database, the measures they use to indicate the usefulness of each database with respect to a given query, and the techniques they employ to estimate the usefulness. In this paper, we study to design an algorithm and compare with the existing algorithm to select the most appropriate search engine with respect to the user query.

Key words: Past Queries, Search engine, Algorithms, Meta Search, Ranking.

INTRODUCTION

Search Engines are widely used for information retrieval, two types of search engines exist. General-purpose search engines aim at providing the capability to search all pages on the Web. Special-purpose search engines, on the other hand, focus on documents in confined domains such as documents in an organization or in a specific subject area11. But any single search engine cannot solve the problem of Internet information retrieval completely because the search engine has limit to search in their own databases.

IR is sub field of computer science concerned with presenting relevant information, collected from web information sources to users in response to search. Various types of IR tools have been created, solely to search information on web3. Apart from heavily used search engines (SEs) other useful tools are deep-web search portals, web directories and meta-search engines (MSEs)3. A person engaged in an information seeking process
has one or more goals in mind and uses a search system as a tool to help achieve those goals.

Metasearch engines are being developed in response to the increasing availability of conventional search engines. The major benefits of MSEs are their capabilities to combine the coverage of multiple search engines and to reach deep web. Search plans are constrained by the resources available: how much time should be allocated to the query and how much of the Internet's resources should be consumed by it. For each search engine selected by the database selector, the component document selector determines what documents to retrieve from the database of the search engine. When user poses a query to the Metasearch through the user interface, the Metasearch engine is responsible to identify appropriate underlying search engine which has relevant document with respect to the user query. Meta search engine selects the appropriate underlying search engine with respect to the user query. 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. To find out the relevant information different similarity measure is used which estimate the relevance between document and user query.

The rest of the paper is organized as: In Section II Meta Search engine (MSE), In Section III Related work, Section IV provides Proposed work, Section V discusses about Experimental result, Section VI discusses Result based comparison, Section VII presents Conclusion & Future scope.

Metasearch engine (MSE)
A Meta Search Engine overcomes by virtue of sending the user's query to a set of search engines, collects the data from them displays only the relevant records. In other words A metasearch engine is a system that provides unified access to multiple existing search engines. Metasearch engine is generally composed of three parts that is Searching request for pre-processing part, Search interface agent part, Search results processing part. There are a number of reasons for the development of a metasearch engine such as:
- Increase the search coverage of the Web,
- Solve the scalability of searching the Web,
- Facilitate the invocation of multiple search engines
- Improve the retrieval effectiveness

Database selector
In many cases a large percentage of the local databases will be useless with respect to the query. Sending a query to the search engines of useless databases has several problems. if we send the query to useless databases also a large number of documents were returned from useless databases, more effort would be needed by the metasearch engine to identify useful documents. The problem of identifying potentially useful databases to search for a given query is known as the database selection problem. The software component database selector is responsible for identifying potentially useful databases for each user query. A good database selector should correctly identify as many potentially useful databases as possible while minimizing wrongly identifying useless databases as potentially useful ones.

Document selector
The component document selector determines what documents to retrieve from the database of the search engine. The goal is to retrieve as many potentially useful documents from the search engine as possible while minimizing the retrieval of useless documents.

Query dispatcher
The query dispatcher is responsible for establishing a connection with the server of each selected search engine and passing the query to it.

Result merger
After the results from selected component search engines are returned to the metasearch engine, the result merger combines the results into a single ranked list. A good result merger should
rank all returned documents in descending order of their global similarities with the user query.

Related work

When a metasearch engine receives a query from a user, it invokes the database selector to select component search engines to send the query to. A good database selection algorithm should identify potentially useful databases accurately. Many approaches have been proposed to tackle the database selection problem\textsuperscript{16}. In paper\textsuperscript{15}, utilizes the retrieved results of past queries for selecting the appropriate search engines for a specific user query. The selection of the search engines is based on the value of relevance between user query and the search engine.

ALIWEB\textsuperscript{11} an often human-generated representative in a fixed format is used to represent the contents of each local database or a site. In WAIS\textsuperscript{11} For a given query, the descriptions are used to rank component databases according to how similar they are to the query. In D-WISE\textsuperscript{11}, the representative of a component search engine consists of the document frequency of each term in the component database as well as the number of documents in the database. Therefore, the representative of a database with \( n \) distinct terms will contain \( n + 1 \) quantities (the \( n \) document frequencies and the cardinality of the database) in addition to the \( n \) terms. The Collection Retrieval Interface Network (CORI-Net)\textsuperscript{11, 15, 6, 8} is carried out using two pieces of information for each distinct term i.e. document frequency and search engine frequency. In gGlOSS, the usefulness of a database is sensitive to the similarity threshold used. As a result, gGlOSS can differentiate a database with Estimating the Similarity of the Most Similar Document. (ESoMSD): is measure indicates the best that we can expect from a database as no other documents in the database can have higher similarities with the query.

Learning based approach\textsuperscript{11} predict the usefulness of a database for new queries based on the retrieval experiences with the database from past queries. The retrieval experiences may be obtained in a number of ways. First, training queries can be obtained in advance (i.e., before the database selector is enabled). The MRDD (Modeling Relevant Document Distribution) approach\textsuperscript{16} is a static learning approach. During learning, a set of training queries is utilized. Each training query is submitted to every component database. From the returned documents from a database for a given query, all relevant documents are identified and a vector reflecting the distribution of the relevant documents is obtained and stored. Specifically, the vector has the format \(<r_1, r_2, \ldots, r_s>\), where \( r_i \) is a positive integer indicating that \( r_i \) top ranked documents must be retrieved from the database in order to obtain \( i \) relevant documents for the query. Savvy Search (www.search.com) is a metasearch engine employing the dynamic learning approach. In Savvy Search\textsuperscript{11, 13} the ranking score of a component search engine with respect to a query is computed based on the past retrieval experience of using the terms in the query. ProFusion approach\textsuperscript{15, 11} is a hybrid learning approach, which combines both static and dynamic learning approach. In the ProFusion approach, when a user query is received by the metasearch engine, the query is first mapped to one or more categories. The query is mapped to a category that have at least one term that belong to the user query. In Relevant Document Distribution Estimation Method for Resource Selection (ReDDE)\textsuperscript{21}, estimate the distribution of relevant documents across the set of available search engines. For making its estimation ReDDE considers both search engine size and content similarity. ReDDE is the resource selection algorithm that normalizes the search engine size.

Proposed work

The propose algorithm is based on \textsuperscript{15, 16} in which the retrieved documents for each past query from all selected search engines and used to calculate the relevance between search engines and respective past query using top k document. The top k document is a merge list of documents from all search engines. Utilizes the retrieved results of past queries for selecting the appropriate search engines for a specific user query. Since there tends to be many similar queries\textsuperscript{21} in a real world federated search system, the valuable information of past queries can help us provide better resource selection results. In this section, we propose a
algorithm \textit{qSim}, to utilize the valuable information to guide the decision of search engine selection. In the algorithm we estimate the value of \( rel (S_j/p_i) \) for all past queries. For an online query, the task is to estimate \( rel (S/q) \) based on the information of \( rel (S/p) \) and \( sim (p/q) \). The proposed algorithm calculate similarity between past query and user query using cosine function that is based on vector space model. The query similarity algorithm\textsuperscript{15} normalize the value of using a constant but if we consider the constant as a variable and change the value of variable, algorithm produce better result that is not addressed by\textsuperscript{15}, proposed algorithm focused on it. The search engines with higher value of \( rel (S_j/q_i) \) being selected by the Metasearch engine.

The steps of proposed algorithms can be described by assuming that there exists a set of past queries, which is denoted by \( P = \{ p_1, p_2, ..., p_n \} \) where \( p_i \) represents the \( i \)th past query, and information resources for an user query \( S = \{ s_1, s_2, ..., s_n \} \). Now generate the ranked list for \( p_i \) past query and calculate \( rel (S/p) \) and then assign scores for the information sources as\textsuperscript{15}. For a user query \( q = \{ q_1, q_2, ..., q_k \} \) where \( q_i \) represent terms in user query, calculate \( sim (p/q) \) as inspired by\textsuperscript{12, 4}

\[
Sim(p, q) = \frac{P \cdot Q}{|P| \times |Q|} = \frac{\sum_{i=1}^{T} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{T} w_{ij}^2} \times \sqrt{\sum_{i=1}^{T} w_{iq}^2}}
\]

Where \( w_{ij} = tf_{ij} \times idf_i \), and \( tf = \max(f_{1j}, f_{2j}, ..., f_{ij}) \). Term Frequency (TF) is the weight of a term \( t_i \) in document \( d_j \) is the number of times that \( t_i \) appears in document \( d_j \), denoted by \( f_{ij} \).

\( idf_i = \log \frac{N}{df_i} \), Inverse Term Frequency (IDF) , \( N \) is the total number of documents in the system and \( df_i \) be the number of documents in which term \( t_i \) appears at least once.

\[
w_{iq} = \left( 0.5 + \frac{0.5f_{iq}}{\max(f_{1q}, f_{2q}, ..., f_{iq})} \right) \times \log \frac{N}{df_i}
\]

In next step we normalized the value of \( Sim(p_i, q) \) using\textsuperscript{15}, the step of calculating \( CutSimq \) if we consider \( (CutSimq= \alpha \times \text{MaxSimq}) \), where \( \alpha \) is 0.8 in\textsuperscript{15} then by changing the value of variable \( (\alpha = 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1) \) we find better result, that is not addressed by\textsuperscript{15}, proposed algorithm focused on it. Now for user query \( q \) and for (each search engine) compute

\[
Rel(s_j/q) = \sum \text{Rel}(s_i/p_i) \times sim(p_i/q)
\]

Ranked the search engine according to the value of \( Rel(s_j/q) \). A larger value of \( Rel(s_j/q) \) is more likely means it contain most relevant documents with respect to the user query \( q \).

**EXPERIMENTAL**

In order to testify the effectiveness of the propose algorithm is simulated in MATLAB 2010b. In this experiment we assume that there are six search engines, five training query and a user query, assume all past queries and user queries having maximum six terms.. The evaluation of result is as follows.

Let us assume the five past query set and a user

<table>
<thead>
<tr>
<th>Table 1: Past query set(Pi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
</tr>
<tr>
<td>P2</td>
</tr>
<tr>
<td>P3</td>
</tr>
<tr>
<td>P4</td>
</tr>
<tr>
<td>P5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: User query(Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

For every past query \( PQ \), Metasearch engine selects the underlying search engine with respect to the training queries and all training query will apply to all selected search engines to retrieve
document with respect to the user query. Finally a single merge list of documents return by search engine for all training query using [15] will be prepared. Similarly apply the user query(q) to all selected search engines by Metasearch and a single merge list of documents return by the six search engines for the user query using [15] will be prepared.

Table 3:

| Relevance matrix between SE<sub>i</sub> and P<sub>i</sub> | 0.6667 | 0.8333 | 0.8333 | 0.8333 | 0.6667 | 0.8333 | 1.0000 | 0.8333 | 0.5000 | 0.8333 | 0.6667 | 0.6667 | 0.5000 | 0.6667 | 0.8333 | 0.6667 | 0.8333 | 1.0000 | 0.8333 | 0.5000 | 0.6667 | 0.8333 | 0.6667 |
|--------------------------------------------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|

Computes rel(S<sub>j</sub> | p<sub>i</sub>) between a search engine and all past queries for top-6 documents on the basis of merge list of all past queries and user query.

Similarity between user query and all past queries can be calculated as in proposed algorithms using cosine similarity

Table 4:

<table>
<thead>
<tr>
<th>Similarity Matrix between P&lt;sub&gt;i&lt;/sub&gt; and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9560</td>
</tr>
</tbody>
</table>

Now we normalize the values of Sim (p<sub>i</sub> | q) by using the value of variable (α = 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1) and calculate the similarity between user query and all search engines as [14, 15].

Table 5:

<table>
<thead>
<tr>
<th>Similarity Matrix between SE and Q on α=0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2701</td>
</tr>
</tbody>
</table>

By arranging Similarity between SE and q in descending order we get the order of selected search engine.

Table 6:

<table>
<thead>
<tr>
<th>Order of selected search engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 2 6 1 3 5</td>
</tr>
</tbody>
</table>

Table 7:

| Similarity between Pi and Q | Sim(S<sub>i</sub> | q) | Order of selected Search engine |
|-----------------------------|-----------------|---------------------------------|
| Q-sim (New)                 | Q-sim (New)     | Q-sim (New)                     |
| 0.9560                      | 0.9444          | 1.2701                          |
| 0.8462                      | 0.5556          | 1.4100                          |
| 0.8242                      | 0.5556          | 1.2644                          |
| 0.7692                      | 0.8889          | 1.5134                          |
| 0.7253                      | 0.5000          | 1.2280                          |
| 1.3908                      | 1.3039          | 5                               |

Result based comparison

The simulation result of a q<sub>S</sub>_Sim algorithm using rank merge list of documents and [15] algorithms is given in the table 7.

Conclusion & future scope

Resource selection is a valuable component for a search system. This paper proposes query similarity algorithm that find
similarity between past query and user query using cosine similarity and implemented in matlab using pdist function, also tested the normalization by changing the value of variable. Similarity is calculated on merge lists of past queries and merge list of user query. Experimental results demonstrated the advantage of proposed qSim to achieve more effective similarity between search engine and user query in comparison to the existing query similarity algorithm15. If large numbers of search engines are selected for the user query the search engine selection problem would be compounded. So, with respect to the user query it is a challenging task to select the relevant search engine among the various search engines.

REFERENCES


