A man-machine meta-search interface, for handling multiple query submissions in web search services

IOANNIS ANAGNOSTOPOULOS, IOANNIS PSOROULAS, CHRISTOS ANAGNOSTOPOULOS, VASSILI LOUMOS and ELEFTHERIOS KAYAFAS
Electrical and Computer Engineering Department, National Technical University of Athens (NTUA)
9, Heroon Polytechniou Str., Zographou, Athens, GREECE

Abstract: Nowadays it is commonly accepted that the web is structured in a chaotic way. Due to the fact that it grows exponentially, it is a difficult task to efficiently locate specific information. This paper presents a User-defined Meta-Search Engine (UMSE), capable to handle multiple queries in web search services. UMSE utilises up to 10 known search engines according to a user-defined ranking, submits its queries in parallel and finally presents the merging results back to the user. The anticipated outcome of using UMSE is that the user gets enhanced amount of information, recording in parallel his search preferences. UMSE also supports history features in order to personalise user queries and facilitate information retrieval.

Key-Words: man-machine system, meta-search engine, query personalisation

1 Introduction

All known search engines such as Yahoo!, AltaVista or Excite follow the standard technique of using a spider – crawler in order to update an index of already stored web pages. However, due to the extremely rapid growth of the web, search engines cannot spider all the new pages at the same time or with the same priority. Besides, the search engines index their ‘attached’ web pages with different algorithms, having as result different response time in updating their directories.

Thus, the user, who uses a specific search engine, may lose some useful information resources returned from another engine. To avoid this, users need to submit and resubmit their queries to multiple search engines, increasing in this way, an already time consuming process [1]. UMSE scopes to provide the user with additional information, without having to know the query language of the other search services. Furthermore, UMSE will store all the results from all the selected engines in a local repository. The innovation in using the proposed machine stands in the fact that, after the first results, the user can easily work off-line and re-define his search engine preferences in order to compare new and old meta-results.

2 Problems in web searching

With the phenomenal growth of the web, most of the search services have accepted the fact that it would be almost impossible to index the entire web. Instead they concentrate on a specialized subset of the web and use ranking techniques to determine which of the web pages to index [2]. A web user is not aware of this problem and when a search service returns to him no relevant results, he will probably understand that such a resource does not exist.

This proves that search engines are not comprehensive, and the user is forced to resubmit his queries over multiple search services to avoid missing potentially useful information. In addition, during this procedure, the user must be aware of the query syntax of every search engine increasing once more an already consuming process. However, since different search engines are largely incompatible and do not allow for interoperability, UMSE solves two major
problems. The query-translation problem and the meta-results merging problem.

2.1 The query-translation problem

A meta-search engine submits queries over multiple search engines. However, as mentioned above the interfaces and capabilities are different in most of the cases. Even the basic query model that the sources support may vary. Some search engines may support the Boolean retrieval model [3], forming a query condition, the sources of which either do or do not satisfy.

Most commercial and well-known search engines also support different variations of the Vector-Space retrieval model [3]. However, even if two different search engines support the Boolean retrieval model often their query syntax differs. Using a meta-search service through UMSE, it is enough to be aware of only one query syntax. The query-translation problem also presents a barrier from the different stemming algorithms or the stop-word lists that are involved in the query model of each search engine. As a result of all this inconsistency, UMSE translates the user query before submitting it in parallel to the defined search engines.

<table>
<thead>
<tr>
<th>Boolean Operator</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>*</td>
<td>term1*term2</td>
</tr>
<tr>
<td>OR</td>
<td>+</td>
<td>term1+term2</td>
</tr>
<tr>
<td>NOT</td>
<td>-</td>
<td>term1-term2</td>
</tr>
<tr>
<td>EXACT</td>
<td># #</td>
<td>#Exact Phrase#</td>
</tr>
</tbody>
</table>

Table. 1 UMSE basic query syntax

In order to perform such a translation, the syntax of each individual search engine was examined in detail and a common query language for the user of UMSE was formed. The basic syntax of the UMSE query-language is depicted in table 1. The current version of UMSE supports complex multiple query translation such as “term1*term2 –term3+term4” or “term1+#Exact Phrase# -term2”. In the first case the proposed meta search engine translates in parallel a query, asking to get results with term1 and term2 without the term3 or term4. In the second case UMSE returns results from the used search services with term1 and an exact matching string without the term2.

2.2 The meta-results merging problem

Search engines, which support the vector-space model for information retrieval, assign a similarity value-score in their indexed sources according to how similar they appear to the given query. However, there are many ways to measure these similarities, and the used ranking algorithms from the search engines are not available to the public. Thus, merging query results from different search engines that probably use different and unknown algorithms is quite of hard task [4],[5]. Furthermore, it is difficult to merge query results even if the compared search engines use the same ranking algorithm. Even if two search engines use the \( t_i \cdot \log \frac{N}{n_i} \) formula for computing similarity weights, it is highly expected that identical documents will have different scores in respect to the given query. This happens because the two logarithmic ratios of the relevant sources contain the query terms, versus the total number \( N \) of the indexed sources differs [3].

In order to deal with this problem, UMSE proposes a merging algorithmic procedure, which will be configured by the user each time he submits a query in parallel. Based in the fact that a web user generally has a strong preference in one or two search services, UMSE allows him to compare those preferred search engines with others. For example, if a user usually submits his queries to Yahoo!, he will probably lose some information source which is not indexed or updated by this service. There is also the possibility to find the requested information after a significant amount of relevant sources. However, the same user will change his preference to Yahoo!, if for example realises that with the same query-syntax, AltaVista returns the requested information in better ranking position. In other words, having solved the query-translation problem, the user compares the requested information, which is retrieved in the \( r \)th rank position of his favourite search engine, with the results presented until the \((r-1)\)th rank position of the second favourite search engine, the results presented until the \((r-2)\)th rank position of the third preferred search engine etc. A history profile for every submitted query is supported by the system, as well as the capability to work off-line comparing the ranking positions of the returned meta-results. Further details about the algorithmic procedure are given in the followings.

3 Description of the system

This section presents the major subsystems as well as the overall architecture of UMSE. The current version of UMSE has been implemented using Java and it utilises AlltheWeb, AltaVista, DirectHit, DMOZ, Excite, Go, Lycos, Northern Light, and Yahoo! as web search services. Figure 1 presents the overall architecture of the proposed system and how its major parts interact, in order to collect, store evaluate and present the meta-results to the user. The main
subsystems of UMCE are described in the next sections.

3.1 The Graphical User Interface (GUI)

Through the GUI, the user expands his search into multiple search services. As mentioned before, query is simultaneously submitted to different search engines. An important aspect of the UMSE GUI includes a choice of which search engines to involve in the search. The current implemented version of UMSE presently provides a choice of 9 search engines.

The GUI of the proposed meta-search engine is presented in figure 2. There are three major tabs namely Multi Query, Statistics and Downloaded URLs. Activating the Multi Download tab, the user can optionally provide his username, in case he wants to personalise his queries. Studies presented that search engines users, usually restrict their queries to a specific subset of topics on the web [6]. Thus, when a user wants to search for a similar topic, he can work offline and search over the already returned meta-results. The user enters his query according to the proposed UMSE unified format, presented in table 1. Finally, he decides upon the number of the involved search services and defines their order of priority. This order of priority is crucial, since it initiates the meta-results controller. In the sequel, user can either submit his query to the selected search services (on-line submission), or search on the local repository over previous downloaded results (off-line submission). This feature improves significantly the response time and the efficiency of the service over the large amount of the disseminated information. By default, UMSE GUI presents to the user his last submitted query to each search engine as presented in figure 2.

Figure 3 present the GUI when the user decides to submit his queries for images. In the current version UMSE uses three search engines (AltaVista, Excite, Yahoo!) while the rest are inactive, since they don’t support queries for images.
3.2 The Query-Transmitter

When a web user decides to submit a query to UMSE, the transmission phase is initiated. This phase begins with simultaneous query submission up to 10 web search services. The transmitter can be considered as an equivalent of the broker in the resource discovery system [7]. It translates the user query to the respective query language of each search engine, creates a URL object, and through an ‘openStream()’ method, sends the results to the meta-results collector. Since UMSE depends on results extracted from other search engines, it does not need to spider or index sources. Instead, it only communicates with the search services. The transmitter needs to understand and query each service in its own interface, integrating varied search interfaces into a Generic format [8]. The user configures the query-transmitter from the GUI, by providing the query and defining how many search engines to query in parallel.

Furthermore, in this phase, UMSE performs a check of resource accessibility such as network bandwidth and connection availability, before making any use of network resources.

3.3 The meta-results Collector

The next step of UMSE is to collect the results returned from every submitted query. As mentioned above, the query transmitter sends to the collector specific streams (html source codes), which correspond to the returned results of the used search services. The collector, which resides in the local server, ‘reads’ the appropriate fields, such as the URLs and the respective description information areas, and stores them in the local repository. This procedure is performed in parallel for each selected search engine and is terminated when all the returned streams are indexed. It must be noticed that indexing the necessary information fields is not an easy task, since each search service publishes its results in a different way. For example, Excite search engine returns a stream, in which an additional URL object must be created for the collection of the URLs. This is due to the fact that the respective search engine is powered by Overture, transmitting an argument, which cannot be immediately resolved from the collector, affecting the collection procedure in terms of time. Finally, the results of the varied search engines are converted and stored in a unified format for further process.

3.4 The Local Repository

As mentioned in the previous paragraphs, the useful information collected from the related query streams is stored in the local repository. For each search service a unique table space is assigned in order to allow the user to view the contents of the local repository. Additionally, when a search query is submitted and results are obtained, UMSE keeps a record of each query instance, in order to provide statistics. These statistics can highlight several search engines features such as user query rates as well as network statistics such as the download rate and the meta-results parsing time (figure 3 and figure 4).

![Fig. 3 Query rate for the used services](image)

Furthermore, the local repository is kept totally independent of the previous subsystems (transmitter-collector), meaning that it can be processed off-line. The user can observe how the meta-results ranking varies according to different search engine priority. For instance, working offline, the user can define a new order of priority and check whether for a particular request, a better ranking position for a specific meta-result is obtained. Finally, it must be noticed that the local repository is responsible for handling all requests for viewing, searching and modifying the meta-results and is also dedicated to handle requests according to user profiles.

3.5 The meta-results Controller

This is the last subsystem, which is dedicated to tackle the meta-results merging problem. As mentioned above, UMSE proposes a merging algorithmic procedure, which will be configured by the user each
time that a query is submitted to multiple search engines.

Fig. 4 Query rate for the used services

Having solved the query-translation problem, the collector ‘extracts’ the required information from all the submitted services in parallel and updates the respective table spaces in the local repository with the meta-results and the user profile information. Then the duplicate URLs are removed from the table space with the (p+1) priority in respect to the table space with the p priority, where p takes values from 1 up to the total number of the used search services.

Then, the merging algorithm compares whether the information retrieved in the rth rank position of search engine with priority p, exists until the (r-1)th rank position of the search engine with priority p+1, then until the (r-2)th rank position of the search engine with priority p+2, and so on. In case that a unique URL is found until the (r-1)th or (r-2)th rank position in the respective search engines, then this URL is assigned as the mth meta-result. If a duplicate field appears in the above sequence, it is ignored. This procedure ends with the assignment of the last meta-result. The number of the meta-results is the total returned results from all the involved search engines, having removed the duplicated fields. In the current version of UMSE the first 40 results from each search service are taken under consideration, since a larger collected amount will increase the system response time. Table 2, depicts the correlation between priority (p) and result rank (r). Furthermore, the following equations describe the meta-result merging algorithm used by UMSE. In equation 1, \( Q^{(p)} \) defines the intersection between the results placed in the r rank position, of the search services with p and (p+1) priority respectively. In case that the compared results are similar then the next meta-result is given by equation 2, while in the other case the next meta-result is provided by equation 3. By this algorithmic procedure the identical results are eliminated.

![Image](56x416 to 280x722)

**Table 2 UMSE basic query syntax**

<table>
<thead>
<tr>
<th>result rank (r)</th>
<th>search engine priority rank (p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>p1 r1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>r</td>
<td>p1 r2</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>K</td>
<td>p1 rK</td>
</tr>
</tbody>
</table>

\[
Q^{(p)} = R\{p_r r_r\} \cap R\{p_{p+1} r_r\} \quad (1)
\]

If \( Q^{(p)} = \emptyset \),

\[
M_i = R\{p_r r_r\} \quad (2)
\]

If \( Q^{(p)} \neq \emptyset \),

\[
M_i = Q^{(p+1)} = R\{p_{p+1} r_r\} \cap R\{p_{p+2} r_r\} \quad (3)
\]

where:

- p: priority of the used search service,
- r: rank position of each result
- N: the maximum number of the used search services,
- K: the maximum number of the results returned from a used search service,
- \( R\{p_r r_r\} \): the result in the rth position of the search service with pth priority,
- \( M_i \): the next meta-result, provided by the UMSE.

Figure 5 depicts a list of meta-results over a specific user query of the following type, "term1*#Exact Phrase# -term2", as described in section 2.1. The Multi Query Results tab in the GUI returns all the gathered results from 6 search engines according to the explained algorithm. The value after the name of the search service with p priority corresponds to results, which cannot be found in search services with higher priority.
4 Conclusions

This paper proposes a user defined meta-search tool, which aims to extract additional information sources in respect to the submitted queries. The user gets enhanced amount of information, recording his search preferences in parallel. Due to the fact that users usually restrict their queries to a specific subset of topics on the web, UMSE provides offline features, over the already processed meta-results. More search engines can be easily added in the proposed architecture, if the query-translation problem is solved for each one.

Future work involves an additional feature, which will estimate the best search engine priority, for a specific meta-result. Although it is considered as a time-consuming procedure, calculating (N-1)! combinations for N search sources, the working offline capability allows this kind of implementation. Heading in this direction, UMSE also scopes to move in a greater level of personalization, using probabilistic user profiles [9].

References: