Expert Agreement and Content Based Reranking in a Meta Search Environment using Mearf*

B. Uygar Oztekin
University of Minnesota,
Dep. of Computer Science,
Army HPC Research Center
oztekin@cs.umn.edu

George Karypis
University of Minnesota,
Dep. of Computer Science,
Army HPC Research Center
karypis@cs.umn.edu

Vipin Kumar
University of Minnesota,
Dep. of Computer Science,
Army HPC Research Center
kumar@cs.umn.edu

ABSTRACT

Recent increase in the number of search engines on the Web and the availability of meta search engines that can query multiple search engines makes it important to find effective methods for combining results coming from different sources. In this paper we introduce novel methods for reranking in a meta search environment based on expert agreement and contents of the snippets. We also introduce an objective way of evaluating different methods for ranking search results that is based upon implicit user judgements. We incorporated our methods and two variations of commonly used merging methods in our meta search engine, Mearf, and carried out an experimental study using logs accumulated over a period of twelve months. Our experiments show that the choice of the method used for merging the output produced by different search engines plays a significant role in the overall quality of the search results. In almost all cases examined, results produced by some of the new methods introduced were consistently better than the ones produced by traditional methods commonly used in various meta search engines. These observations suggest that the proposed methods can offer a relatively inexpensive way of improving the meta search experience over existing methods.

General Terms

Algorithms, performance, experimentation

Keywords

Merging, reranking, meta search, collection fusion, expert agreement

1. INTRODUCTION

With the current rate of growth of the Web, most search engines are unable to index a large enough fraction of the available web pages. Furthermore, it is becoming increasingly difficult to keep up with the rate at which already indexed resources are updated. Heuristics used in different search engines are often different from each other emphasizing some aspects, de-emphasizing others, not necessarily sustaining the same quality across varying types of queries. Meta search engines have the potential of addressing these problems by combining search results from multiple sources. They can provide better overall coverage of the web than provided by any individual search engine. They can also offer potentially better overall rankings by taking advantage of the different heuristics that are used by different search engines.

A key component of a meta search engine is the method used to merge the individual lists of documents returned by different engines to produce a ranked list that is presented to the user. The overall quality of this ranking is critical, as users tend to examine the top ranked documents more than the lower ranked documents. There are many meta search engines available on the web ([18], [17], [2], [12], [15], [10]), but due to commercial nature of most of these systems, technical details of the underlying collection fusion methods are often unavailable. Most of the meta search engines, for which technical details are available ([4], [20], [23]), use a variation of the linear combination of scores scheme (LC) described by Vogt and Cottrell [24]. This scheme requires that a weight is associated with each source (reflecting its importance) as well as a weight associated with each document reflecting how well it matches the query. Then, it uses a product of the two to compute an overall score for each document to be used in ranking. If the weight of each source is unknown or uniform and if the sources only provide a ranked list of documents but no numerical scores, which is the case for most search engines, then this scheme becomes equivalent to that of interleaving the ranked documents produced by the different sources.

The focus of this paper is to study different methods that can be used to merge the results in the context of meta search engines. To this end, we introduce four novel methods for merging results from different search engines and evaluate their performance. The schemes we are proposing are motivated by the observation that even though the various search engines cover different parts of the web and use different ranking mechanisms, they tend to return results in which the higher ranked documents are more relevant to the query. Presence of the same documents in the results of different search engines in top ranks can be a good indication about their relevance to the query. In fact some existing LC-based methods already use this observation to boost the ranks of such documents. The methods we are proposing take advantage of this observation, and also look for common themes present in top ranked documents to extract a signature that can be used to rerank other documents. As a result, un-

*Available at http://mearf.cs.umn.edu/

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

like LC-based methods, our methods can boost the ranks of the documents that are similar in content to the top ranked documents deemed relevant. These new methods that use expert agreement in content to merge and rerank documents have been incorporated in our meta search engine, Mearf, which is accessible from http://mearf.cs.umn.edu/.

We experimentally evaluated the rankings produced by these methods with two variations of the linear combination-based approaches that are commonly used in many meta search engines. Our experimental evaluation was based on a systematic analysis of the query logs from Mearf over the course of a year, involving over ten thousand distinct queries. We propose an evaluation method that uses implicit user judgements seen through user clicks and introduce average position of clicks as a metric that can be used in evaluating different methods automatically and objectively under certain conditions. Although far from perfect, this approach arguably offers better statistical significance than what can be practically achieved by explicit user feedback.

Our experiments show that the choice of the method used for merging the output produced by different search engines plays a significant role in the overall quality of the search results. In almost all cases examined, results produced by some of the methods introduced were consistently better than the ones produced by traditional LC-based methods commonly used in various search engines. As a reality check, we also compare our methods to Google, a popular and highly regarded search engine used by Mearf, to see whether or not the results produced by Mearf methods as well as LC-based methods contain more relevant documents that appear earlier in the ranked-list presented to the users. Our results show that LC-based methods do not perform better than Google, but some of the Mearf methods are consistently better than Google according to the evaluation criteria. These observations suggest that the proposed methods can offer relatively inexpensive way of improving the meta search experience over existing methods.

The remainder of the paper is organized as follows: Section 2 presents related work, Section 3 gives an overview of the Mearf architecture, Section 4 gives a detailed description of the fusion methods implemented in Mearf with runtime analysis, Section 5 presents the experimental setup and discusses the results obtained, and finally Section 6 summarizes the results presenting conclusions and suggestions for future research.

2. RELATED WORK

Metacrawler [20, 21, 17] is probably one of the first meta search engines that were developed in the context of the world-wide-web. Its architecture was similar to current meta search engines and used a relatively simple mechanism to combine the results from the different search engines, eliminating duplicate URLs and merging the results in an interleaving fashion possibly taking into account scores returned if available. Profusion [23, 18], another early meta search engine, shares some of the characteristics of Metacrawler, but employs a somewhat more sophisticated approach for combining the results. In this approach, each search engine has a confidence value assigned with it, and each document returned by the search engine has a score assigned to it that is normalized between zero and one. This score is taken either directly from the search engine's output or is derived from the ranked list. Profusion then multiples these two scores (search-engine confidence and document score), and ranks the documents in decreasing order of this score. Later publication on Metacrawler [19] suggests that, at some point, it too used a linear combination based scheme called Normalize-Distribute-Sum algorithm, similar to Profusion's approach. Savvy Search [4, 9] focuses primarily on the problem of learning and identifying the right set of search engines to which to issue the queries, and to a lesser extent on how the returned results are merged. Callan, et al. [3] have focused on building a framework to find collections containing as many relevant documents as possible and suggested three ranking schemes depending on different scenarios: (i) if no rankings are available, then use interleaving; (ii) if scores from different collections are comparable, then use the score from the sources to produce a global ranking; (iii) if scores are not comparable, then use a weighting scheme to form a global ranking by calculating the weights associated with each collection. Finally, Inquirus [10] took an entirely different approach to the problem of combining the results of different search engines. Instead of relying on the engine's ranking mechanism, Inquirus retrieves the full contents of the documents returned and ranks them more like a traditional search engine using information retrieval techniques applicable to full documents only (*e.g.*, using cosine similarity to query, extracting information about query term context and proximity of query terms in the documents, etc.). This approach can potentially offer better ranking at the cost of scalability. Recent additions to Inquirus include topic-geared query modification and ranking incorporated in an incremental user interface [7]. Besides these widely known non-commercial meta search engines, a number of meta search engines are available [2, 12, 15]. However, due to their commercial nature, there is limited information on the underlying approaches used to combine the results.

In general, most of the known methods used for combining the results of different search engines in the context of meta search can be classified as a variation of the linear combination of scores scheme (LC), generalized by Vogt and Cottrell [24]. In this approach, the relevance of a document to a query is computed by combining both a score that captures the quality of each source and a score that captures the quality of the document with respect to the query. Formally, if q is a query, d is a document, s is the number of sources, and $\mathbf{w} = (w_1, w_2, \ldots, w_s)$ are the source scores, then the overall relevance ρ of d in the context of the combined list is given by

$$\rho(\mathbf{w}, d, q) = \sum_{systems} w_i \rho_i(d, q)$$

In the context of meta search engines, this translates to assigning weights to each one of the search engines and a weight to each link (using the score of the link in the search engine if available, otherwise using a function of the rank to obtain a score), multiplying the two to obtain the final score for each link (just it was done in the case of Profusion). Linear combination of scores approach is also used in various information retrieval systems to query distributed databases or to combine different retrieval approaches and query representations from a single database (e.g. [22, 1, 16]).

Besides the above directly related research, the underlying techniques used in meta search engines also draw ideas from a number of different areas of classical information retrieval,

including source selection, information fusion, reranking, and presentation. In the rest of this section we briefly review some of the most relevant research in these areas.

The problem of source selection focuses on identifying the right collections to be queried given a particular user query. In the context of meta search engines, source selection can be used to select what subset of meta search engines to use. This is especially useful in the context of specialized queries. Gravano et al. [8] assumed that they have access to term frequency information for each database and proposed to use this information and the query terms to obtain an estimate of how much relevant documents each source would return for a given query. French et al. [5, 6] proposed metrics for evaluating database selection techniques and compared the two approaches. Wu, Yu, Meng, et al. [26, 25] proposed an efficient source selection method that could be used when the number of databases are fairly large. They also have a nice summary of major components in meta searching especially regarding source selection and collection fusion problems. Query probing methods have been proposed [11] to obtain approximate statistics about sources such as term frequencies by sending a number of query probes, enabling methods based on term frequency and other information of the sources to be used up to a degree in situations in which one does not have access to the documents in the collection nor their statistics.

The problem of collection fusion is focused towards selecting the best sources and how many items to be retrieved from each so as to maximize coverage under restrictions on the number of items that will be retrieved. Most of the work related to collection fusion problem for distributed databases is not directly applicable to Web meta search context. Majority of the search engines supply data in predetermined increments like 10 or 20 links, and a typical user rarely examines more than a few tens of links for a given query. Specifying a fractional amount of links to be retrieved from each search engine is feasible but due to practical considerations and the incremental nature of the results, current systems tend to use all of the links that are retrieved from a particular search engine. It is also not common practice to adjust the number of links to be retrieved from a search engine based on the query terms. In general the server selection in a meta search environment is a binary or at most a discrete problem (e.g., select 20, 50, or 100 links from a particular search engine). Methods based on document scores are not directly applicable to meta search domain either: Majority of search engines do not report any scores at all, and there are considerable variations between the ones reporting some sort of scores.

3. MEARF FRAMEWORK

Mearf is a typical meta search engine augmented with text processing abilities. Its user interface consists of a CGI program that allows to input a query string, and select a subset of the supported search engines.

Various modules in Mearf and their interaction are summarized in Figure 1. Once a query is submitted to Mearf, its search engine interface module connects to the selected subset of search engines, obtains their results in the html format, parses them, removes advertisements, and extracts and returns the actual links. Note that if the number of links to be retrieved is larger than a given search engine's link increment value, multiple html pages are retrieved until

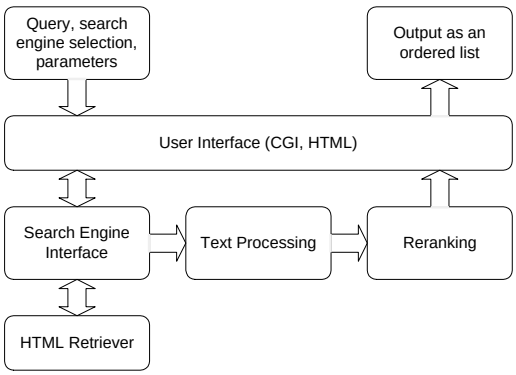

Figure 1: Mearf structure

either search engine's results are depleted or until the requested number of links are fetched. The search engine interface module basically handles all communications to the search engines. To expedite retrieval, it opens multiple connections via multi-threading. For each link, the associated URL, URL title, and snippet information are passed on to the text processing module. This module processes URL titles and snippets (stop list, stemming, tf-idf normalization) to form a sparse vector for each link to be used in vector-space model operations in the reranking module. Once the duplicates are removed and a reranking method is applied, the results are presented to the user.

We have a dictionary consisting of about 50K stemmed words and an augmented stop list, both geared for html and snippet domain. If a term does not appear in the dictionary but appears in the query, it is assumed to be a rare term and gets assigned a predetermined important idf value, if a term is neither in the query nor in the dictionary, or if it is in the stop list but not in the query, it is ignored. We normalize each vector using 2-norm. We implemented sparse vector, html and text processing modules handling all html to text conversions, word stemming (a variation of Porter's stemming algorithm), text to sparse vector conversions, and operations on sparse vectors. All of these modules, including multithreaded html retrieving and search engine interfacing, are written in C++, balancing efficiency, flexibility and ease of maintenance.

Mearf uses a robust duplicate removal scheme. It is able to detect a very large majority of duplicate URLs and/or mirrors with very few false positives. Although it is possible to merge different URLs if their snippets are very similar (for example, updated version vs older version of the same web page), these cases are rare, and benefits of a decent duplicate removal scheme outweighs the loses.

With this framework we are able to extract and process hundreds of links per search engine, and most search engines are willing to supply a maximum of 300 to 800 links to a regular user. By using five to ten search engines, Mearf architecture can quickly retrieve and process up to a few thousands of unique links for a given query. Default behavior of Mearf for a regular user is to forward the query to four or five search engines, and retrieve from each 20 links. For a typical query, after the duplicates are removed, we are left with about 60 to 70 unique links. Unlike traditional search engines that display results in increments, Mearf presents

Figure 2: Mearf results for the query "C++ stl programming"

all in a single, compact, ordered list fitting about 20 results on a typical browser page.

Our objective behind Mearf was to build an experimental testbed that will allow us to evaluate different fusion approaches. To generate enough traffic and to encourage people to use Mearf, we advertised it in our department homepage by putting a small search box, forwarding the query to Mearf's own page (http://mearf.cs.umn.edu/). Mearf was publicly available since November 2000 and has a small but stable user base, attracting several hundred queries every week from users worldwide. Figure 2 shows how the user-interface looks like for a typical user.

Mearf has two modes: standard mode and superuser mode (which is activated via a hidden cgi parameter). In standard mode, a user is only allowed to select which search engines to use, but he/she has no control on any other parameters. Once a regular user types a query and hits the "go" button, Mearf issues parallel queries to its set of search engines and then randomly selects one of the six different fusion methods implemented to rerank the results. We also added another method that uses only Google with its original rankings into the randomly selected methods pool, i.e., some of the queries that the user makes in Mearf is nothing more than a search using Google. Note that in order to evaluate the different fusion methods in an unbiased way, the standard interface of Mearf does not allow the user to specify or know which one of the different methods is used in reranking the results. In standard mode, for each query, Mearf records a number of statistics, including the query text itself, the fusion method that was randomly selected, as well as the ranks of the returned documents that were clicked on (if any) by the user. In superuser mode, which is only used by the members of our group, additional diagnostic information is presented, and the user has control on all of the Mearf parameters including the method selection, but in this mode, no statistics are recorded. Members of our group used Mearf strictly in superuser mode, hence none of the queries we made affected the logs that are used in the evaluations.

4. RERANKING METHODS

Unlike many meta search engines, fusion methods used in Mearf do not solely rely on the original scores and/or the order of the links returned by the search engines. Mearf implements six different methods for fusing together the results of the different search engines. The first two methods, called Interleave and Agreement, implement variations of widely used linear combination of scores approach ([24], [23], [20], [22], [1], [16]). We used up to four or five general purpose, popular search engines and assigned them equal weights. Note that if different weights are available for different search engines, Interleave and Agreement methods can be easily modified accordingly. In the remaining four methods, namely Centroid, WCentroid, BestSim, and BestMSim, we first find a set of relevant documents and rerank all the documents based on their cosine similarity to a vector obtained from the relevant set. Original rankings do play a role in the selection of the relevant set but the set produced for each method is different.

The key motivation behind the Mearf methods is that the different search engines can be thought of as experts, and the set of documents that they returned can be considered as their expert answers on the particular query. The key assumption is that answers for which different experts agree on are more likely to be relevant than answers for which there is little agreement among experts.

Let us first introduce some of the notation used in describing the methods, then we will explain each method, starting from the naïve ones.

4.1 Notations

In search engine results, a link (or a document) consists of a triplet (url, url title, snippet). In Mearf we augment this with a sparse vector by processing the url title and the snippet. Thus a link forms a quadruple (url, url title, snippet, vector). We will use the notation l_i^s to denote the i^{th} link from search engine s and $vector(l)$ to denote the sparse vector of link l.

A permutation $p(pos_{s1}, pos_{s2}, \ldots, pos_{sn})$ of size n is defined to be an n-tuple of positive integers, and an entry pos_{si} denotes the position of a link from search engine i, where n is the number of search engines used in the query. For example if we have four search engines the permutation $p(1, 6, 5, 3)$ states that we selected the 1^{st} link from search engine 1, 6^{th} link from search engine 2, 5^{th} link from search engine 3, and 3^{rd} link from search engine 4.

A range selection $rs(set_{s1}, set_{s2}, \ldots, set_{sn})$ of size n applied to permutations of size n is used to put a limit on the allowed permutations of size n for a given context.

Each set_{si} is a set of positive integers and a permutation $p(pos_{s1}, pos_{s2}, \ldots, pos_{sn})$ restricted with a range selection $rs(set_{s1}, set_{s2}, \ldots, set_{sn})$

is valid only if $\forall i, (i \in [1, n] \wedge i \in N) \Longrightarrow pos_i \in set_i$ where N is the set of positive integers.

Note that the number of valid permutations for a given range selection $rs(set_{s1}, set_{s2}, \ldots, set_{sn})$ is:

$|set_{s1}| \times |set_{s2}| \times |set_{s3}| \times \cdots \times |set_{sn}|$ where $|set_i|$ denotes the cardinality of set set_i.

The $rank$ of a particular link in a given search engine, is the position of the link in the results for that search engine. We will use the notation $score(l)$ to denote the relevance measure of a link l calculated by Mearf using one of the methods, higher values denoting better relevance. Score of

a link is a real number in the range $[0, 1]$ in all but one method (in the Agreement method, the possible range is theoretically $[0, n]$, where n is the number of search engines, maximum value occurring only if all n search engines report the same URL in their first position).

The $+=$ operator is used to denote addition then assignment, e.g., if a and b are two variables (or vectors) $a+=b$ denotes $a = a + b$. For a vector v, $|v|_2$ denotes the second norm of vector v.

4.2 Interleave

Interleaving is probably the first method one might think of in information fusion. In this method, we interleave the results coming from different search engines, visiting result sets of search engines one by one for each rank: take the firsts from all search engines, seconds from all, thirds from all and so on. If the current link from a search engine is a duplicate of a previously visited link, we skip this link and go on to the next search engine. Note that in this method, duplicate links are reported only when the first occurrence is seen. If the individual rankings of the search engines are perfect and each search engine is equally suited to the query, this method should produce the best ranking. Interleave method corresponds to linear combination of scores scheme [24] with equal server weights, taking the best score in case of duplicates. The following pseudo-code outlines one possible implementation:

```
let n be the number of links to be retrieved from each engine
let results be an empty array of links
for i = 1 to n
    for s = 1 to number_of_search_engines
        if l_i^s exists and is not a duplicate of links in results
            insert l_i^s at the end of results.
return results
```

4.3 Agreement

In the Interleave method, if a link occurs in multiple search engines, we selected the best rank and ignored the others. However, one might suggest that a link occurring in multiple search engines can be more important than the ones occurring in just one engine at similar ranks. For instance a link that has 3^{rd}, 2^{nd}, 2^{nd}, and 3^{rd} ranks in four different search engines, respectively, may be a better link than the one that has 1^{st} or 2^{nd} rank in one search engine only. To improve the rankings of this type of documents, we implemented the "Agreement" scheme that is described in the following pseudo-code.

```
let results be an empty array of links
for each link l_i^s
    score(l_i^s) = [1/rank(l_i^s, s)]^c
while there are duplicate links across search engines
    merge the links by adding up their scores
add all links to results
sort links in results according to their scores
return results
```

Where c is a parameter that can be used to control how much boost a link will get if it occurred multiple times. As an example: if c is 1, a link occurring at 4^{th} rank in two search engines will have a score of $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$ making it equal in score to a link occurring at 2^{nd} position, and better than any link occurring at 3^{rd} position in a single search engine only. If c were 0.5, if we do the same calculation: $\frac{1}{2} + \frac{1}{2} = 1$, we see that now it has precedence against 2^{nd} and higher placed single links. If c is small, emphasis on agreement is increased, if it is large, this effect is reduced. Another way to control how agreement affects the results might be to use $c = 1$ but give different weights to duplicate ranks according to the number of duplicates across search engines. Yet another way, is to adjust weights according not only to the number of duplicates but also to the ranks of each duplicate. In our current implementation we are adding up the ranks with parameter c set to 1. Note that this method is very similar to linear combination of scores scheme [24] with equal server weights, if scores of the duplicates are added.

4.4 Centroid

We developed two methods based on centroids: Centroid and WCentroid (weighted centroid). In both of them, the key idea is that the first k links coming from each search engine can be trusted to be relevant to the query. In the Centroid method we find the average (or centroid) of the vectors of the first k links reported from each search engine and then rank all the links using the cosine measure of its vector to the centroid vector calculated.

```
let k be the number of top links to be considered in ranking
let centroid be an empty sparse vector
let results be an empty array of links
for s = 1 to number_of_search_engines
    for i = 1 to k
        if l_i^s exists
            centroid = centroid + vector(l_i^s)
centroid = centroid/|centroid|_2
for each link l_i^s
    score(l_i^s) = vector(l_i^s) · centroid
while there are duplicate links across search engines
    merge duplicates by taking the maximum of the scores
add all links to results
sort links in results according to their scores
return results
```

4.5 WCentroid

The previous method did not consider rank of the links in the search engines. Another approach is to weight the links according to the place of the links in the search engines. The first links will be given higher weights, and we will decay the weights of the links according to their place in top k. We used a linearly decaying weighting function starting with 1 at the 1^{st} rank, and min_val at the k^{th} rank, where min_val is a value between 0 and 1. If min_val is set to 1, it becomes equivalent to the Centroid method. We suggest a value between 0.25 and 0.5, if k is small (about 5), and a value between 0 and 0.25, if k is larger. Although we tried non-linear weighting functions, we found this approach to be simple and effective for the ranges of k used in Mearf.

```
let k be the number of top links to be considered in ranking
let centroid be an empty sparse vector
let results be an empty array of links
for s = 1 to number_of_search_engines
    for i = 1 to k
        if l_i^s exists
            centroid+ = vector(l_i^s) · [1 − ((i−1)·(1−min_val))/k]
centroid = centroid/|centroid|_2
for each link l_i^s
    score(l_i^s) = vector(l_i^s) · centroid
while there are duplicate links across search engines
```

merge duplicates by taking the maximum of the scores
add all links to $results$
sort links in $results$ according to their scores
return $results$

Weighted centroid method can be considered as a method using a relevance set with each item weighted differently according to some criteria instead of being treated equally. In our case the weights are obtained according to the ranks of the links in the search engine they are coming from.

4.6 BestSim

The two centroid methods used search engines' own rankings in selecting the relevant set used in reranking. BestSim and BestMSim schemes use a slightly different approach. We still consider the top k links from each search engine but the relevant set is not all the first k links, but a subset of them selected according to the contents of the links. In BestSim method, we try to find a link from each source so that the tuple of links selected has the maximum self-similarity.

More formally, we consider all permutations $p(pos_{s1}, pos_{s2}, \ldots, pos_{sn})$ restricted with a range selection $rs(\{1,2,\ldots,k\},\{1,2,\ldots,k\},\ldots,\{1,2,\ldots,k\})$, and try to find the best permutation $bp(r_1, r_2, \ldots, r_s)$ where the self-similarity of the vectors of the links $l^1_{r_1}, l^2_{r_2}, \ldots, l^s_{r_s}$ is the highest among all possible permutations.

The rationale behind both BestSim and BestMSim methods is to use expert agreement in content to select the relevant set.

let $current\_best = -1$
for each search engine i
  $set_i = \{1, 2, \ldots, min(k, number\_of\_links\_returned(i)\}$
if all set_is are empty
  return nil
for each valid permutation $p(r_1, r_2, \ldots, r_s)$
under $rs(set_1, set_2, \ldots, set_s)$
  $centroid = \sum_{i=1}^{s} vector(l^i_{r_i})$
  if $|centroid|_2 > current\_best$
    $current\_best = |centroid|_2$
    $best\_centroid = centroid$
$best\_centroid = best\_centroid/|best\_centroid|_2$
for each link l^s_i
  $score(l^s_i) = vector(l^s_i) \cdot best\_centroid$
while there are duplicate links across search engines
  merge duplicates by taking the maximum of the scores
add all links to $results$
sort links in $results$ according to their scores
return $results$

4.7 BestMSim

This method is similar to BestSim method, but instead of looking for a single permutation with best self-similarity we try to find the first m best permutations. In the beginning we consider the first k links from each search engine, find the permutation with highest self-similarity, record it, remove the links selected from candidate sets, and then augment them by the next available links $(k+1)$. After doing this m times, we obtain the relevance set. Note that, in our implementation, a link from each search engine can only appear in one of the permutations. For instance, let us suppose that we start with 5 links from each search engine (links 1,2,3,4,5) and select the 1^{st} from 1^{st} engine, 3^{rd} from 2^{nd} engine, and 5^{th} from 4^{th} engine. For the second iteration, we will consider links numbered 2,3,4,5,6 from first engine, 1,2,4,5,6 from the second one, 1,2,4,5,6 from the third one and so on in selecting the next best similarity. We continue until we find m tuples or run out of links.

let $ranking\_vector$ be an empty sparse vector
for $i = 1$ to s
  $set_i = \{1, 2, \ldots, min(k, number\_of\_links\_returned(i)\}$
for $i = 0$ to $m-1$ (*)
  for each valid permutation $p(r_1, r_2, \ldots, r_s)$
  under $rs(set_1, set_2, \ldots, set_s)$
    $centroid = \sum_{i=1}^{s} vector(l^i_{r_i})$
    if $|centroid|_2 > current\_best$
      $current\_best = |centroid|_2$
      $best\_centroid = centoid$
      for $j = 1$ to s
        $index[j] = r_j$
    for $j = 1$ to s
      $set_j = set_j - \{index[j]\}$
      $set_j = set_j + \{(k+i)\}$
    $ranking\_vector += best\_centroid/|best\_centroid|_2$
$ranking\_vector = ranking\_vector/|ranking\_vector|_2$
for each link l^s_i
  $score(l^s_i) = vector(l^s_i) \cdot ranking\_vector$
while there are duplicate links across search engines
  merge duplicates by taking the maximum of the scores
add all links to $results$
sort links in $results$ according to their scores
return $results$

A variation for BestMSim is to weight the vectors of links in each permutation among the m permutations found according to the self-similarity measure of the permutation, giving higher emphasis on more coherent permutations. Yet another approach is to use a decaying weighting function assigned to each permutation number, first one getting a weight of 1, and linearly decaying the weights up to the m^{th} permutation, analogous to centroid - weighted centroid schemes, decaying the weights as i in loop (*) increases.

In some sense, BestSim method can be considered as a method capturing the main theme present in the first k results from each search engine. We feel that it could be more suited for specific queries. BestMSim, on the other hand, has the potential to capture more than one theme in the first $k+m$ links. Thus, it may be preferable in the case of multi-modal or general queries.

4.8 Runtime Analysis

In all Mearf methods, we find a relevant set, remove duplicates, and rerank all documents according to the reranking vector found from the relevant set. The only difference in costs between these four methods are in the selection and processing of the relevant set to form the reranking vector.

A reasonable duplicate removal scheme can be implemented with runtime costs in the range from $O(n \log n)$ to $O(n^2)$, where n is the total number of links retrieved. The lower-bound corresponds to a unique sort, with the possibility to use either the URL strings or a processed version of them as keys. (In Mearf we have a notion of URL stemming. We have a few rules to stem various prefixes as well as postfixes to better handle redundant naming schemes with very few false positives. For instance, "www.cs.umn.edu/~oztekin/",

"www.cs.umn.edu/~oztekin", "cs.umn.edu/~oztekin", and "www.cs.umn.edu/~oztekin/index.html", are all mapped to the same key.) This scheme takes care of the majority of the duplicate URLs, but it may also be desirable to identify and remove mirrors. In this case, we combine the previous approach with pairwise similarity comparison. If both the titles and the body of the snippets are very similar, one may identify them as mirrors, possibly taking into account a token-wise URL similarity if desired. $O(n^2)$ run time is due to the mirror identification part. It may also be possible to find a compromise in between, balancing robustness and runtime.

Once the reranking vector is found, ranking the results and sorting them takes $O(n)$ and $O(n \log n)$ time, respectively.

Let us now investigate the time required to obtain the reranking vector in all four methods. Assuming that we have s search engines, investigate top k links from each engine, and for the case of BestMSim, find m best tuples. In our implementation, cost of forming the reranking vector using Centroid and WCentroid methods is $O(sk)$, with BestSim, it is $O(k^s)$, and finally, for BestMSim, it is $O(mk^s)$.

Note that for small s, k and m compared to n, which is the case for Mearf (their values are set to 4 or 5), the cost of finding the duplicates (ranges from $O(n \log n)$ to $O(n^2)$ depending on the method used) and ordering the results ($O(n \log n)$) dominates the runtime in the case of Centroid and WCentroid methods. If n is sufficiently large compared to other parameters, then this is also valid for BestSim, and BestMSim. In fact comparing the six methods in terms of processing time, the difference between Interleave and Agreement methods vs four Mearf methods is minimal (not detectable in most cases with our resolution of ~20 milliseconds). If we look at the total runtime for a query, network connection time takes about 2 seconds, total processing time including parsing the cgi parameters, all text processing, reranking, html generation, and logging takes about 0.1 to 0.3 seconds under light load (our server is not purely dedicated to Mearf).

5. EXPERIMENTAL EVALUATION

5.1 Methodology

One way of evaluating the results of different fusion methods is to select a number of users and a set of queries and let the users explicitly judge the performance of these methods on the selected set of queries. This method can give fairly accurate results on the performance of the methods for that particular set of queries. But due to practical reasons, only a small fraction of the possible queries as well as users can be sampled. When we deviate from this approach, we do not have explicit information about relevant and non-relevant set of documents for a representative set of queries. If we had, we could directly use them to evaluate different methods using traditional information retrieval approaches. Evaluation methods that use implicit relevance information have been proposed as an alternative in the lack of explicit judgements. One such method uses automated ways to simulate user judgements, typically using measures such as cosine similarity between the query and the documents, and term frequencies and/or phrase frequencies of the query terms present in the text [14]. Even though this approach has the potential to sample a wider range of queries, top ranked results returned by a typical search engine are already expected to have the query terms in relatively high frequencies since search engines rank the results with similar methods. Thus, objective applicability of this approach to our domain is limited.

Another way of evaluating the performance of the methods is to judge them by the implicit relevance indication as seen in the user logs. This approach enables us to span all types of queries submitted to Mearf, as well as the whole range of users who issued them, providing much greater statistical significance, but it has its own drawbacks. We only have information about the documents that the users have clicked on for each query, i.e., we know the positions of the links investigated for each query. Although the fact that a user deciding to investigate a set of links by clicking on them can show a good indication about the relevance of the summary (snippet and title in our case), it does not necessarily show that the actual documents referred to are relevant to the query. Nevertheless, when good snippets are used to describe the documents, we believe that the correlation is reasonably high. It is also possible that a document can be highly relevant to the query but has poor snippet and title. Other cases are also possible (user clicks on the link but the document is no longer online, etc.) For all of the cases, one may argue that with a large enough sample, these cases will be evenly distributed among the methods and will not particularly favor or disfavor one method against others.

In our evaluations, we have chosen the last approach and used the logs produced by Mearf during the course of almost a year (11/22/2000 to 11/10/2001). In the very beginning, the search engines used by Mearf were Altavista, Directhit, Excite, Google and Yahoo!, but Yahoo! was soon eliminated as it started to use Google's technology. Table 1.a summarizes the overall characteristics of the data set obtained from the logs. Table 1.b shows the characteristics of the data for different fusion methods. The column labeled "avg results per query" is the average number of documents returned by Mearf for each query, the column labeled "number of queries" is the number of times a particular method was selected to rerank the results, the one labeled "number of clicks" shows the total number of documents that were clicked using the corresponding method, and the column labeled "click ratio" is the number of times a particular method is used which resulted in at least one user click, divided by total number of times the method is used in reranking. Note that for some methods, the number of times they were used is smaller than the rest. This is mainly because we designed and introduced the methods incrementally in the beginning. In addition, we removed Interleave and Agreement methods from the random pool after five months once we had enough samples to be confident that they were inferior to others. This allowed us to focus on our newly introduced methods and compare them better against each other in various scenarios.

5.2 Metrics

For a typical query, average user scans through the returned list, in general starting from the top ranks, and clicks the links that are apparently relevant to what the user was searching for. If the snippet or the title of a page does not seem interesting, typical user quickly skips it without clicking on the link. This process goes on until one or more satisfactory documents are found or he/she gets bored or

1.a High level statistics	
total number of queries	17055
number of queries with clicks	10855
number of clicks	34498
average clicks per query	2.02
avg clicks per query ignoring queries without clicks	3.18
click ratio (queries with clicks / total number of queries)	0.64
average retrieval time	1.99 sec
average processing time	0.29 sec
average total time per query	2.28 sec

1.b Statistics for each method				
method	avg results per query	number of queries	number of clicks	click ratio
Interleave	62.64	1530	3015	0.64
Agreement	62.09	655	1241	0.60
Centroid	61.74	3381	6702	0.64
WCentroid	61.70	2403	5018	0.65
BestSim	61.93	3443	6817	0.62
BestMSim	61.45	3220	6671	0.65
Google	48.25	2423	5034	0.64

Table 1: Overall characteristics of the dataset

decides to augment or change the query. We believe that a good search engine or meta search engine using a list presentation should order the links according to relevance. The above observations suggest that the performance of the ordering of a reranking method could be implicitly measured by looking at the position of the links that the users found interesting and clicked on. Intuitively, if the user selects k specific links and investigates them, a method that places these k links in higher ranks (preferably first k positions) is superior to a method that does not.

For all methods (except the one that directly retrieves Google results), given a query, we retrieve the same set of links. Since we are focusing on the reranking aspects, it became natural to consider metrics that primarily take ordering into account. In evaluating the ordering of various methods, we tried a few approaches including average position of the clicks (the lower, the better), average position of the clicks normalized with the number of links retrieved, and uninterpolated average precision (in the range 0 to 1, 1 corresponding to the perfect case) as discussed in [13]. We also considered a few variations removing outliers (e.g., positions 50 and higher, typically occurring very infrequently) and/or redundant duplicate clicks (same click for the same session occurring multiple times). As an illustration, let us suppose that the user selects the 5^{th} link for method A, and 8^{th}, 9^{th}, and 10^{th} links for method B. Let us also consider that the same amount of links, say 20, are returned for both cases. The average position of clicks are 5 and 9, respectively, which makes method A superior to method B using this metric. The uninterpolated average precision on the other hand, is 0.2 for method A, and about 0.216 for method B, making method B better than method A according to this metric. Given roughly the same amount of total links presented, we found that the average position of the links that a user clicked on is a more intuitive, easy to interpret, and relatively unbiased way in comparing two methods. Removing outliers and duplicates did not make a significant difference, and we chose to remove duplicate clicks only (some users may double click where only one click is sufficient, or due to slow connection and slow update, they can click on the same link multiple times). Normalizing the average positions of the clicks with the total number of clicks was not desirable, since, looking at the histogram of the clicks, we felt that it could introduce bias.

While calculating average position of clicks, we ignored the queries which resulted in no clicks. One might argue that the ratio of the queries with clicks vs total number of queries for a particular method should also be considered in comparing the methods. For instance, if users choose not to click any of the results returned more for a particular method compared to others, this may be an indication that the method is not producing desirable results. However, we did not see a significant difference in the value of click ratio among different methods in overall data and its various subsets. Table 1.b, last column, shows the ratio for each method.

We selected the average ranks (positions) of clicks as the metric used in the evaluations, lower values showing that the clicks occur in top ranks in the list, and higher values showing that the clicks occur in lower portions in the list. We would like to point out that since all documents are returned in a single page, user tends to scroll down the page easily and lookup for a relevant document even at lower ranks. Hence, if the number of returned documents is larger, the average rank of the clicked documents also tends to be higher. This trend, clearly visible in Table 3, holds for all fusion methods as well as Google. Note that this metric focuses on the relevance of the snippets and titles in the eyes of the users. We assume that a typical user, in general, makes his decision to click or not to click a particular document based on the perceived relevance of the title and the snippet. We also assume that the relevance of the snippets and titles on the average are positively correlated to the relevance of the documents. In the following sections, when we say that a method is better than another method, what we mean is that the method is better in placing more relevant snippets in better positions compared to the other method according to the users' implicit judgements. If the relevance of the snippets and titles are highly correlated to the relevance of the documents, this would further suggest that a better method in this metric will also be a better method in supplying better documents in higher positions. If summaries are not available or if the correlation does not hold for a particular domain, the four Mearf methods are still applicable if the full documents are used, but evaluating them in that domain may require different approaches.

In order to be able to compare two methods using the average position of the clicks, number of links returned by the two methods should roughly be the same except maybe in the case in which the method having the smaller average of the two also has more links than the other. For example, if 20 links are returned for one method, and 50 for another, with average ranks of clicks of 5 and 15, respectively, it is unclear which of the two methods is superior to the other. On the other hand, if the average rank of clicks were 15 and 5, respectively, we could argue about the second method being superior to the first one (since for the second method, on the average, user finds an interesting document at 5^{th} link out of 50, compared to 15^{th} out of 20 for the first method). This issue is not a problem in comparing the first six methods implemented among themselves since the number of unique links returned on the average is roughly the

method	AvgRank	StdevRank
Interleave	17.37	18.68
Agreement	17.39	19.72
Centroid	12.74	14.36
WCentroid	12.88	14.19
BestSim	13.64	14.85
BestMSim	13.57	15.16
Google	13.90	15.16

Table 2: Overall performance of methods

same for all. But for Google's case, this was not true. We sampled the logs, calculated the mean and standard deviation of the number of links returned on the average for the six remaining methods, and set Google method to ask for a uniformly distributed number of links accordingly. This approach works fine for general queries, but for specific queries, Google returns considerably fewer number of links compared to the number of links requested, as well as the total number of links that would be retrieved from multiple search engines for the same query. This trend is clearly visible in first two sub-tables of Table 3, where statistics only contain the queries which returned 0 to 24 links and 25 to 49 links. In the column labeled "Clicks" in these tables, we can see that Google has considerably more samples than others, especially in the 0 to 24 links range.

5.3 Results

Table 2 summarizes the overall performance of the six fusion methods implemented as well as the results produced by Google. The column labeled "AvgHits" shows the average number of links retrieved for that method, the column labeled "AvgRank" shows the average position of the documents that the user deemed as relevant by clicking on them, and the column labeled "StdevRank" shows the standard deviation of the positions of the relevant documents.

The histogram in Figure 3 presents the overall behavior of methods in a finer level of granularity. In particular, the x-axis corresponds to the rank of the relevant documents (documents that are clicked) using a bin size of 5, and the y-axis corresponds to the fraction of the relevant documents that fall within a particular bin. For example, the very first bar in the first bin indicates that about 37% of the documents using the Interleave scheme that the users clicked on were placed by the Interleave method in the top 5 positions in the final sorted list. As can be expected, we have more clicks in first bin for all methods, and fraction of the clicks drop down as we go to bins corresponding to higher ranks. Note that the first two bars of each bin correspond to Interleave and Agreement methods respectively; the next four bars correspond to Mearf methods: Centroid, WCentroid, BestSim, and BestMSim, in this order; and finally, the last bar corresponds to Google. In the first three bins, the convex shape of top of the columns in the bins suggests that the fraction of the clicks of Mearf methods are higher compared to other methods in these bins. Later on we see that it gradually transforms to a concave shape in subsequent bins, suggesting that Mearf methods have fewer clicks in these bins.

Looking at the overall results in Table 2, we can see that the centroid based schemes do better than the rest in the sense that they rank the relevant documents higher. The BestSim and BestMSim schemes are somewhat worse than

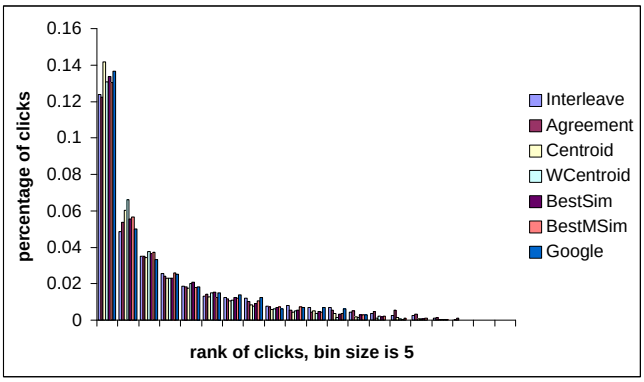

Figure 3: Histogram of ranks of clicks for different methods.

the two centroid based schemes, but better than the rest. Comparing Google against the centroid and best-sim-based methods, we can see that all four Mearf methods does better than Google despite the fact that the number of links returned were considerably fewer for Google on the average (˜48 vs ˜62). Note that just like the example given in section 5.2, we cannot draw conclusion about the relative performance of Interleave and Agreement methods with respect to Google, since the number of links returned for Google were significantly smaller than the others. This is further investigated in the next paragraph.

We segmented the data set according to number of links returned and examined each case separately in a finer level of detail. These results are shown in Table 3. The first sub-table contains only the queries that returned up to 24 documents, the second contains queries that returned 25–49 documents, the third contains queries that returned 50–74 documents, and the last one contains the remaining queries. Columns labeled "AvgHits" is the average number of links returned for that method, and "Clicks" is the number of clicks used in the statistics for that method. Examining these results, we can see that the centroid based schemes are better than all other schemes including Google for all cases except the first one (0–24 links). In this case, drawing objective conclusions is difficult since average number of links retrieved for the methods are relatively different (varying from 5.73 to 11.37), and the number of clicks, i.e., samples that the statistics are based on, are much smaller for the first group (ranges from 28 to 410) than for the other groups (where it ranges from few hundreds to several thousands). Behavior of all methods in general, may have less statistical significance for this case than other cases. BestSim and BestMSim also perform better than the Interleave and Agreement schemes as well as Google for all but the first group. Since we did not retrieve more than 74 links from Google, there are no Google statistics for the last subtable. Looking at these results with comparable number of links returned, Google, Interleave, and Agreement methods does not have a clear winner among the three, but they are consistently outperformed by the remaining four methods except the first sub-table.

Next, we analyzed the results with respect to the length of the queries performed by the users. Table 4 presents results obtained by the different fusion methods by considering the

0-24 links returned				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	11.37	103	5.47	4.80
Agreement	8.31	28	4.68	4.06
Centroid	11.23	158	4.94	4.79
WCentroid	11.35	98	6.63	7.29
BestSim	9.71	123	4.28	4.27
BestMSim	9.65	153	6.20	5.65
Google	5.73	410	5.31	5.35

25-49 links returned				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	38.80	221	11.27	10.99
Agreement	39.48	126	12.11	11.91
Centroid	40.40	534	10.24	10.12
WCentroid	41.76	455	10.05	10.36
BestSim	40.12	544	11.12	10.55
BestMSim	40.20	487	10.00	9.93
Google	40.26	645	11.41	10.84

50-74 links returned				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	64.07	1658	16.49	17.27
Agreement	64.97	594	15.63	17.24
Centroid	64.13	4340	12.78	14.03
WCentroid	64.34	3301	12.56	13.77
BestSim	64.25	4461	13.46	14.37
BestMSim	64.44	4273	13.55	14.89
Google	59.50	3979	15.19	16.08

75+ links returned				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	81.21	1033	21.28	21.74
Agreement	80.29	493	21.59	23.38
Centroid	79.44	1670	14.18	16.46
WCentroid	78.22	1164	15.45	16.43
BestSim	79.38	1689	15.63	17.16
BestMSim	79.28	1758	15.24	17.09
Google	n/a	n/a	n/a	n/a

Table 3: Comparison of methods with varying number of links returned

queries of length one, two, three, four and greater than four. These results suggest that Centroid and WCentroid methods generally perform reasonably good with varying number of terms in the query. One interesting trend we found is that although BestMSim performs better than BestSim for small number of terms, it gets worse with the increased number of terms, and for greater than 3 terms, BestSim begins to outperform BestMSim. Since the fraction of the data for these queries is relatively small, this can be a spurious pattern. Nevertheless one possible explanation for this behavior is as follows: Queries with small number of terms tend to be general or multi-modal in nature. For these queries, BestMSim is more suitable as the relevant set computed by this scheme may contain many distinct documents. Queries with large number of terms on the other hand tend to be more specific. For such queries, the main topic may be captured by the first (or first few) documents, and the documents selected by BestMSim after the first few iterations may not be very related to the query.

Finally, in evaluating the results produced by either one of the four proposed fusion schemes, we noticed that Mearf has a natural way of filtering out bad links. Our experiments with a large number of randomly selected general queries shows that, for a good majority of these queries, at least the bottom 10% links did not contain any of the query terms, nor a closely related term, whenever the ranking is done using one of the four Mearf methods. It can be difficult to subjectively judge the relative importance of highly ranked links produced by Mearf compared to other search engines. However for a typical query, once we look at the bottom 10% of results produced by Mearf and the position of these links in the original search engines, we see that these mostly irrelevant links are scattered all around in the original search engine, not necessarily in the bottom 10%. Although there is no guarantee that the bottom 10% of the links in Mearf are all irrelevant, these links will be omitted by most users, as their snippets typically do not contain the query terms nor any related terms. Mearf consistently places broken links, links with poor snippets, links with generally irrelevant snippets such as "no summary available", "document not found", "under construction" and snippets with very few terms to bottom 10%, while populating top ranks with snippets containing the query terms and related terms.

6. CONCLUSION AND FUTURE WORK

We introduced four new methods for merging and reranking results in a meta search environment that use content-based agreement among the documents returned by different search engines. All of the four methods, centroid based methods in particular, provide an inexpensive and automated way of improving the rankings. We also introduced a metric that can be used to compare different reranking schemes automatically based upon implicit user judgements seen through user clicks. This metric is applicable when a user can reasonably judge the relevance of a document by looking at the summary provided by the search engine.

Experimental results suggest that selection of methods or adjustment of parameters can be done on the fly based on the number of terms in the query and the number of results returned. For example, the experiments discussed in Section 5.3 indicate that the parameter m in BestMSim can be adjusted depending on the number of terms in the query. For a larger number of query terms, a smaller value of m is expected to perform better. It is also possible to develop hybrid of some of these methods as well as incorporate additional quality measures. None of our methods relies on servers to report scores associated with each link, but our framework is flexible enough to incorporate both server scores and link scores as well as other quality measures (e.g., the snippet length, past user judgements on the URLs, as well as originating domains, etc.) quite easily if available. In fact, we successfully incorporated one such measure in all four Mearf methods, to gradually penalize very short snippets.

Although Mearf architecture is able to retrieve and process a fairly large number of links, due to practical considerations, we did not let regular users have control on the number of links that will be retrieved per search engine, but set it to 20. Most search engines were able to supply fairly relevant links in this range. It may be interesting to extend this study to larger return sets, and compare these methods and others in varying situations. Some of the methods can be more suitable for some ranges, but not for others.

All queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	62.64	3015	17.37	18.68
Agreement	62.09	1241	17.39	19.72
Centroid	61.74	6702	12.74	14.36
WCentroid	61.70	5018	12.88	14.19
BestSim	61.93	6817	13.64	14.85
BestMSim	61.45	6671	13.57	15.16
Google	48.25	5034	13.90	15.16

1 term queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	53.10	501	16.26	17.73
Agreement	51.57	218	19.49	21.93
Centroid	53.61	955	12.81	14.33
WCentroid	53.29	757	11.43	13.12
BestSim	54.25	1035	12.83	14.29
BestMSim	51.82	939	12.71	14.38
Google	48.80	697	11.61	13.95

2 term queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	65.75	1101	17.75	18.85
Agreement	65.96	334	17.93	21.25
Centroid	63.62	2332	13.24	15.25
WCentroid	63.71	1663	13.70	14.75
BestSim	63.95	2347	14.88	16.01
BestMSim	64.22	2412	12.50	13.96
Google	50.48	1936	15.79	16.00

3 term queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	65.89	684	16.43	17.75
Agreement	65.09	302	17.61	19.19
Centroid	65.42	1586	12.46	13.83
WCentroid	64.31	1305	13.51	14.44
BestSim	64.26	1689	13.14	13.93
BestMSim	64.67	1548	13.35	15.06
Google	49.28	1128	12.99	14.40

4 term queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	66.11	391	18.42	19.98
Agreement	66.82	145	15.86	17.01
Centroid	64.04	1004	12.50	13.91
WCentroid	66.49	631	11.82	13.82
BestSim	64.11	941	13.80	15.01
BestMSim	64.59	950	15.33	16.38
Google	46.40	718	13.06	14.76

5+ term queries				
method	AvgHits	Clicks	AvgRank	StdevRank
Interleave	65.95	338	18.46	19.56
Agreement	65.31	242	15.42	17.18
Centroid	63.88	825	12.10	13.24
WCentroid	65.37	659	12.33	13.53
BestSim	66.06	805	11.98	13.35
BestMSim	64.69	822	16.04	17.48
Google	38.97	555	13.10	14.89

Table 4: Comparison of methods with varying query length

By combining different methods and dynamically adjusting their parameters, it may be possible to build better methods suitable in a wider range of situations. For example, a hybrid of WCentroid and Agreement may produce a suitable method if we have a fairly large number of results from each search engine and the histogram of the relevance of the documents vs their positions may exhibit a significant drop after some point. Such a method will boost the rankings of the links that have summaries similar to the reranking vector computed by WCentroid, while respecting the original rankings produced by the search engines up to a degree.

7. REFERENCES

[1] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *Research and Development in Information Retrieval*, pages 173–181, 1994.

[2] C4.com. http://www.c4.com/.

[3] J. P. Callan, Z. Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.

[4] Daniel Dreilinger and Adele E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, 1997.

[5] James C. French and Allison L. Powell. Metrics for evaluating database selection techniques. In *10th International Workshop on Database and Expert Systems Applications*, 1999.

[6] James C. French, Allison L. Powell, James P. Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the performance of database selection algorithms. In *Research and Development in Information Retrieval*, pages 238–245, 1999.

[7] E. Glover. *Using Extra-Topical User Preferences to Improve Web-Based Metasearch*. PhD thesis, 2001.

[8] L. Gravano, H. García-Molina, and A. Tomasic. The effectiveness of GIOSS for the text database discovery problem. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 23(2):126–137, June 1994.

[9] Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[10] Inquirus. http://www.inquirus.com/.

[11] Panagiotis Ipeirotis, Luis Gravano, and Mehran Sahami. Automatic classification of text databases through query probing. Technical Report CUCS-004-00, Computer Science Department, Columbia University, March 2000.

[12] Ixquick. http://www.ixquick.com/.

[13] D. D. Lewis. Evaluating and Optimizing Autonomous Text Classification Systems. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, Washington, 1995. ACM Press.

[14] Longzhuang Li and Li Shang. Statistical performance evaluation of search engines. In *WWW10 conference*

posters, May 2–5, 2001, Hong Kong.

[15] Mamma. http://www.mamma.com/.

[16] M. Catherine McCabe, Abdur Chowdhury, David A. Grossman, and Ophir Frieder. A unified environment for fusion of information retrieval approaches. In *ACM-CIKM Conference for Information and Knowledge Management*, pages 330–334, 1999.

[17] Metacrawler. http://www.metacrawler.com/.

[18] Profusion. http://www.profusion.com/.

[19] E. Selberg. *Towards Comprehensive Web Search*. PhD thesis, 1999.

[20] E. Selberg and O. Etzioni. Multi-service search and comparison using the MetaCrawler. In *Proceedings of the 4th International World-Wide Web Conference*, Darmstadt, Germany, December 1995.

[21] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, (January–February):11–14, 1997.

[22] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *Third Text REtrieval Conference*, 1994.

[23] Mario Gomez Susan Gauch, Guijun Wang. Profusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science*, 2(9):637–649, 1996.

[24] Christopher C. Vogt and Garrison W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[25] Zonghuan Wu, Weiyi Meng, Clement Yu, and Zhuogang Li. Towards a highly-scalable and effective metasearch engine. In *WWW10 Conference, May 2–5, 2001, Hong Kong*. ACM, 2001.

[26] Clement T. Yu, Weiyi Meng, King-Lup Liu, Wensheng Wu, and Naphtali Rishe. Efficient and effective metasearch for a large number of text databases. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2-6, 1999*, pages 217–224. ACM, 1999.