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

B. Uygar Oztekin, George Karypis, Vipin Kumar
University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
{oztekin, karypis, kumar}@cs.umn.edu

June 4, 2001

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. Our experimental evaluation shows that some of our methods produce rankings that are consistently better than the rankings produced by methods that are commonly used in many meta search engines as well as rankings produced by a popular search engine.

Keywords: 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 may be dramatically different from each other, emphasizing some aspects, de-emphasizing others, not necessarily sustaining the same ranking quality across varying types of queries. Meta search engines have the potential of addressing all these problems, as they combine search results from many different search engines. 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 fusion method that merges the individual lists of documents returned by different search engines to produce a ranked list that is presented to the user. The overall quality of this ranking is critical, as most users tend to click on the top ranked documents ignoring a large number of lower ranked documents. The problem of combining search results from multiple sources is referred to as the collection fusion problem in information retrieval and some work in this area is related to meta search ([1], [3], [2], [7]).

There are a large number of meta search engines available on the web. 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, use a variation of the linear combination of scores scheme described by Vogt and Cottrell [8]. This scheme assumes availability of a weight for each source (reflecting its importance) and a score for each document produced by a source. It uses a product of the two to compute an overall score for each document and an overall ranking. If the weight of each source is unknown (or is considered equal) and if the sources only provide a ranked list of documents, (but no

*A long version of this paper including detailed discussion of related work and more thorough analysis of the results with examples is available as a techical report at http://www.cs.umn.edu/

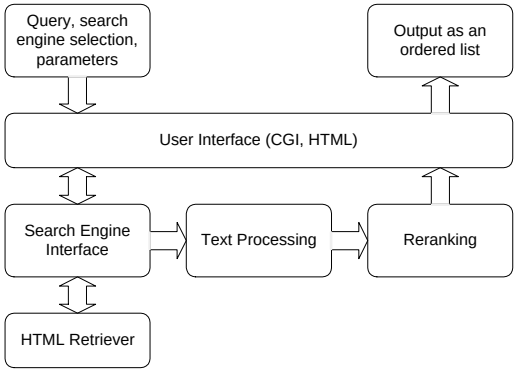

Figure 1: Mearf structure

numerical scores), then this scheme becomes essentially equivalent to one that simply interleaves the ranked documents produced by different sources. A few minor variations of this schemes are possible depending upon how the duplicate documents (i.e., documents present in the output of multiple sources) are handled.

In this paper, we introduce four novel methods for fusing results from different search engines. A set of documents that are individually considered to be important by different search engines and are similar to each other (in content) are likely to be quite important. Furthermore, any other documents that are similar (in content) to these documents are also important. Our experience with traditional search engines suggests that engines that have different coverages and use different ranking heuristics may produce drastically different results even among the top 10 to 20 documents retrieved. However, for most of the queries, many of the top ranked documents are usually relevant to the query. Our methods analyze a small number of top ranked documents for each search engine to find a relevant set of documents and use them to rerank all other documents.

These new methods have been incorporated in our meta search engine, Mearf, which is accessible from http://mearf.cs.umn.edu/. To allow reasonable response time, our implementation deals only with the snippets returned by the search engines and not the complete documents. We compare the ranking produced by these methods with those produced by commonly used interleaving based methods as well as Google. We experimentally evaluated the different methods by performing a user study over the course of seven months. Our experiments show that the proposed methods produce rankings that are consistently better than the rankings produced by methods that are commonly used in many meta search engines as well as rankings produced by Google.

The remaining of the paper is organized as follows: Section 2 gives an overview of the Mearf architecture, Section 3 gives a detailed description of the fusion methods implemented in Mearf, Section 4 presents and discusses the experimental results, and finally Section 5 summarizes the results and presents the conclusions.

2 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.

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 either search engine's results are depleted or until the requested number of links are retrieved. 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 is 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, results are presented to the user.

We have a dictionary consisting of about 46,000 stemmed words and an augmented stop list, both geared for html and snippet domain. If a term does not appear in our 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 very 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. Default behavior of Mearf 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 80 unique links. We believe that most users would not go beyond this number for most queries.

3 Ranking Methods

Unlike many meta search engines, the 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 existing schemes([8], [6], [5]). In the remaining four methods, called 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 new methods implemented in Mearf 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 agrement among experts.

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

3.1 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 $p(pos_{s1}, pos_{s2}, ..., pos_{sn})$ of size n is defined to be an n-tuple of positive integers, and an entry pos_{si} denotes a 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^{th} link from search engine 4.

A range selection $rs(set_{s1}, set_{s2}, set_{s3}, ..., set_{sn})$ 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 $p(pos_{s1}, pos_{s2}, ..., pos_{sn})$ restricted with a range selection $rs(set_{s1}, set_{s2}, set_{s3}, ..., set_{sn})$

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 for a given range selection $rs(set_{s1}, set_{s2}, set_{s3}, ..., set_{sn})$, the number of valid permutations is:

$|set_{s1}| \times |set_{s2}| \times |set_{s3}| \times ... \times |set_{sn}|$ 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).

3.2 Interleave

Interleaving is probably the simplest method one might think in information fusion. In this method, we just 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 [8] with equal server weights, taking the best score in case of duplicates. The following pseudo-code outlines the Interleave method.

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 $results$

3.3 Agreement

In the interleaving 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 compared to the ones occurring in just one 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 in controlling how much boost a link will get if it occurred multiple times. As an example: if c is 1 and a link occurring at 4^{th} rank in two search engines, will have a rank 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 the linear combination of scores scheme [8] with equal server weights, in which scores of the duplicates are summed.

3.4 Centroid

We developed two methods based on centroids: Centroid and W(eighted)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 rank all the links using the cosine measure of its vector to the centroid vector calculated.

let k be the number of top links that should 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
    $score(l_i^s) = vector(l_i^s) \cdot centroid$
while there are duplicate links across search engines
    merge duplicates by taking the maximum of the scores of duplicates
add all links to $results$
sort links in $results$ according to their scores
return $results$

3.5 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 that should 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
            $centroid = centroid + vector(l_i^s) \cdot [1 - \frac{(i-1)\cdot(1-min\_val)}{k}]$
$centroid = centroid/|centroid|_2$
for each link l_i^s
    $score(l_i^s) = vector(l_i^s) \cdot centroid$
while there are duplicate links across search engines
    merge duplicates by taking the maximum of the scores of duplicates

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 rank of the links in the search engine they are coming from.

3.6 BestSim

The two centroid methods used search engines' rankings in selecting the relevant set used in reranking. BestSim and BestMSim schemes use a slightly different approach. We also 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 are trying to find a link from each source so that the tuple of links selected has the maximum self similarity. More formally:

We are considering all permutations $p(pos_{s1}, pos_{s2}, ..., pos_{sn})$ restricted with a range selection $rs(\{1,2,...,k\},\{1,2,...,k\},...,\{1,2,...,k\})$, and find the best permutation $bp(r_1, r_2, ..., r_s)$ where the self-similarity of the vectors of the links $l^1_{r_1}, l^2_{r_2}, ..., l^s_{r_s}$ is maximum 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
    $set_i = \{1, 2, ..., min(k, number\_of\_links\_returned(i)\}$
if all set_is are empty
    return nil
for each valid permutation $p(r_1, r_2, ..., r_s)$ under $rs(set_1, set_2, ..., set_s)$
    $centroid = \sum_{i=1}^{s} vector(l^i_{r_i})$
    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 of duplicates
add all links to $results$
sort links in $results$ according to their scores
return $results$

3.7 BestMSim

This method is similar to BestSim method, but instead of looking for a single permutation with best self similarity we are trying 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 augment them by the next available links $(k+1)$. After doing this m times, we obtain the relevance set. Note that 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 selected the 1^{st} from 1^{st} engine, 3^{rd} from 2^{nd} engine, 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
    $set_i = \{1, 2, ..., min(k, number\_of\_links\_returned(i)\}$
for $i = 0$ to $m - 1$ (*)
    for each valid permutation $p(r_1, r_2, ..., r_s)$ under $rs(set_1, set_2, ..., set_s)$
        $centroid = \sum_{i=1}^{s} vector(l^i_{r_i})$
        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
            $set_j = set_j - \{index[j]\}$
            $set_j = set_j + \{(k + i)\}$
        $ranking\_vector = ranking\_vector + best\_centroid/|best\_centroid|_2$
$ranking\_vector = ranking\_vector/|ranking\_vector|_2$
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 of duplicates
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 similarity measure of the permutation, this approach gives better emphasis on more coherent permutations. Yet another approach is to use a decaying weight 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.

4 Experimental Methodology and Results

4.1 Methodology

Although it is possible to compare the results produced by different methods visually by trying different queries, we felt that any subjective method would not be conclusive in comparing the methods. For this reason we evaluated the performance by studying how users interact with the fusions produced by different schemes. In particular we evaluated the different schemes by comparing the ranks of the documents in the final list that the user found relevant. To determine the relevance of a document we used the implicit relevance indication provided by whether or not the user clicked (i.e., decided to view) a particular document. Even though this approach is far from perfect, previous research [4] has shown it to be reasonably accurate, especially when good snippets are used to describe the documents.

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/). While in Mearf, the user is only allowed to select which search engines to use, but he/she has no control of any of the parameters in Mearf. Once a user types a query and hits the "go" button, Mearf issues parallel queries to the different search engines and then randomly selects one of the different fusion methods described in Section 3 to rank the results. The number of documents requested from each of the search engines was fixed to 20. The results are displayed back to the user in a single page, using a compact presentation fitting about 20 links in a typical browser page. 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 to rank the results. To evaluate the quality of the rankings produced by Mearf,

total number of queries	9377
number of queries with clicks	5746
number of clicks	18998
avg clicks per query	2.03
avg clicks per query ignoring queries without clicks	3.31
avg retrieval time (seconds)	2.17
avg processing time	0.38
avg total time per query	2.55

1(a)

method	avg number of results per query	number of queries	number of clicks
Interleave	62.64	1530	3015
Agreement	62.09	655	1241
Centroid	62.37	1817	3537
WCentroid	62.66	871	1912
BestSim	62.37	1941	3754
BestMSim	61.86	1771	3788
Google	46.33	792	1751

1(b)

Table 1: Overall characteristics of the dataset

we added another "fusion" scheme that uses only Google with its original rankings. That is, some of the queries that the user makes in Mearf is nothing more than a search using Google.

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 actually clicked (if any) by the user. In the following evaluations, we used the logs produced by Mearf from a seven month period (11/22/00 - 5/16/01). In the beginning, the search engines used by Mearf were Altavista, Directhit, Excite, Google and Yahoo but later on, we eliminated Yahoo as it started using Google's technology. Table 1(a) summarizes the overall characteristics of the data set obtained from Mearf's logs. Table 1(b) shows the characteristics of the data for different fusion methods. The column labeled as "avg number of 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 fusion method was selected to fuse the results, and the column labeled "number of clicks" shows the total number of documents that were clicked from the resulting document set. Note that for some fusion schemes, the number of times they were used is smaller than the rest (e.g. WCentroid, Agreement, Google). This is because these methods were introduced towards the middle of the seven month period.

Note that the average number of results per query is much smaller for Google than for other methods. The reason is that, in the beginning we retrieved 50 documents per query from Google, which was considerably smaller than the average number of documents returned by the other methods. During the course of our experiments we realized that due to the design of Mearf (i.e. all the results returned in a single page) the total number of documents returned has a significant impact on the average ranks of the documents that were clicked. Since all documents are returned in a single page, the user tends to scroll down the page easily and lookup for a relevant document. Hence if the number of returned documents is larger, then 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. We selected a sample from the top 100 queries, we calculated the mean and standard deviation of the number of links returned, and set Google method to retrieve a uniformly distributed number of links accordingly. However as the results in Table 1 shows, the average number of documents returned by queries with Google is still quite small as it is biased by the results obtained in the early stages of the experiment and the fact that even with the modification the average number of documents returned is smaller that what it is asked.

4.2 Results

Table 2 summarizes the overall performance of the six fusion methods implemented by Mearf as well as the results produced by Google. The column labeled "AvgRank" shows the average rank of the documents that the user deemed as relevant (using the clicks as an indirect relevance indication), and the column labeled "StdevRank" shows the standard deviation of the ranks of the relevant documents.

The histogram in Figure 2 presents these results in a finer level of granularity. In particular, the x-axis corresponds to the rank of the relevant documents (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 indicates that about 12.2% of the documents using the Interleave scheme that the users found relevant were placed by the Interleave method in the top 5 positions in the final sorted list.

method	AvgRank	StdevRank
Interleave	17.37	18.68
Agreement	17.39	19.72
Centroid	13.23	15.11
WCentroid	13.31	14.44
BestSim	14.19	15.42
BestMSim	14.68	16.16
Google	14.59	15.51

Table 2: Overall performance of methods

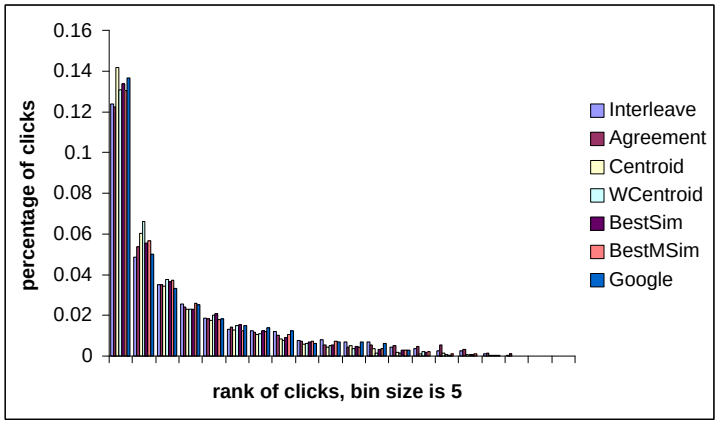

Figure 2: Histogram of ranks of clicks for different methods.

Looking at the overall results we can see that the centroid based schemes do better than the rest in the sense that they rank the relevant documents higher, and that the Interleave and Agreement based schemes do considerably worse. The BestSim and BestMSim schemes are somewhat worse than the two centroid based schemes, but not by a large factor. Comparing Google against the centroid and best-sim-based methods we can see that it does somewhat worse than the centroid schemes and BestSim, but a bit better than the BestMSim. As discussed earlier, these results are biased in favor of Google, as initially our system was returning fewer documents for Google than for the rest of the fusion methods. For this reason we segmented the entire data set into four groups based on the number of documents that were returned. 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 contains the remaining queries.

The difference is small for the first group (0-24 links returned) but tends to be larger for the other three groups. Also note that the number of clicks for the first group are much smaller (ranges from 28 to 143) than for the other groups (where it ranges from few hundreds to several thousands). Hence, the relative superiority of the centroid based schemes has greater statistical significance for these cases. Looking at results in Table 3, we can see that the centroid based schemes are substantially better than all other schemes including Google for all four cases. Note that BestSim and BestMSim also perform better than the Interleave and Agreement schemes as well as Google for all but the first group. As we discussed earlier, there are no Google queries that returned more than 74 documents.

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 queries of length one, two, three, four and greater than four. These results suggest that centroid and weighted centroid based 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 after 3 terms, BestSim begins to outperform BestMSim. Since

0-24 links returned				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	11.37	103	5.47	4.80
Agreement	8.31	28	4.68	4.06
Centroid	10.79	94	4.55	4.69
WCentroid	10.78	29	4.17	4.39
BestSim	9.80	93	4.66	4.56
BestMSim	10.74	114	5.88	5.08
Google	5.97	143	4.73	4.91

25-49 links returned				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	38.80	221	11.27	10.99
Agreement	39.48	126	12.11	11.91
Centroid	39.87	279	9.77	9.76
WCentroid	41.08	186	10.01	9.31
BestSim	38.91	305	11.02	10.63
BestMSim	39.37	248	11.06	10.53
Google	43.67	338	12.04	11.58

50-74 links returned				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	64.07	1658	16.49	17.27
Agreement	64.97	594	15.63	17.24
Centroid	64.46	2023	13.18	14.44
WCentroid	64.51	1085	12.60	13.63
BestSim	64.32	2169	13.93	14.82
BestMSim	64.65	2194	14.58	15.67
Google	59.66	1270	16.37	16.66

75+ links returned				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	81.21	1033	21.28	21.74
Agreement	80.29	493	21.59	23.38
Centroid	80.80	1141	14.87	17.34
WCentroid	79.91	612	16.01	16.71
BestSim	80.54	1187	16.23	17.49
BestMSim	80.53	1232	16.40	18.11
Google	n/a	n/a	n/a	n/a

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

our data set 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 multi-modal. For these queries, BestMSim is more suitable, as the relevant set computed by this scheme contain many distinct documents. Queries with large number of terms tend to be very specific. For such queries, the main topic is 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 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 judge the relative importance of highly ranked links produced by Mearf relative 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 bottom 10%. Although there is no guarantee that the bottom 10% of the links produced by Mearf are all irrelevant but most of the users would omit these links anyway if they do not see any query terms or related terms in the snippets. Mearf consistently places broken links, links with poor snippets, links with irrelevant snippets such as "no summary available", "404 not found", "under construction" and snippets with very few terms to bottom 10%, while populating top ranks with snippets highly related to the query and related terms with possibly better coverage compared to a single search engine.

5 Conclusion

We introduced four novel methods for fusing the results in a meta search environment that use content-based agreement among the documents returned by different search engines. Our experiments, and subjective evaluations showed that these methods can be used to improve the rankings of a meta search engine considerably in an automated manner. All of these methods, especially centroid methods and BestSim provide an automated and inexpensive way of improving the rankings by using only the snippets. We also examined the performance of the methods with queries of different sizes. The results suggest that selection of methods or adjustment of parameters could be done on the fly based on the number of terms and number of results returned. For example, parameter m in BestMSim can be adjusted depending upon the number of terms in the query.

Note that none of our methods rely on servers to report scores associated with each link although our

All queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	62.64	3015	17.37	18.68
Agreement	62.09	1241	17.39	19.72
Centroid	62.37	3537	13.23	15.11
WCentroid	62.66	1912	13.31	14.44
BestSim	62.37	3754	14.19	15.42
BestMSim	61.86	3788	14.68	16.16

1 term queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	53.10	501	16.26	17.73
Agreement	51.57	218	19.49	21.93
Centroid	53.28	453	12.57	14.87
WCentroid	53.95	260	11.28	11.92
BestSim	54.48	583	14.08	15.28
BestMSim	51.90	503	12.98	14.40

2 term queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	65.75	1101	17.75	18.85
Agreement	65.96	334	17.93	21.25
Centroid	64.98	1322	13.64	15.85
WCentroid	66.60	654	14.02	15.05
BestSim	64.89	1209	15.27	16.78
BestMSim	65.09	1309	13.25	14.39

3 term queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	65.89	684	16.43	17.75
Agreement	65.09	302	17.61	19.19
Centroid	65.99	739	13.05	15.26
WCentroid	63.51	483	14.51	15.48
BestSim	64.94	952	13.63	14.28
BestMSim	65.12	902	14.37	16.25

4 term queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	66.11	391	18.42	19.98
Agreement	66.82	145	15.86	17.01
Centroid	65.30	573	13.05	14.14
WCentroid	67.76	239	13.18	15.75
BestSim	64.07	580	14.55	15.35
BestMSim	64.50	570	17.07	17.77

5+ term queries				
method	avg hits	clicks	AvgRank	StdevRank
Interleave	65.95	338	18.46	19.56
Agreement	65.31	242	15.42	17.18
Centroid	63.13	450	13.19	13.95
WCentroid	64.68	276	11.55	11.34
BestSim	66.66	430	12.09	13.77
BestMSim	65.83	504	17.92	19.04

Table 4: Comparison of methods with varying query length

framework is flexible enough to incorporate both server scores and link scores as well as other quality measures if available. Not all methods are best suited for each and every situation. It may be possible to develop hybrid schemes that combine features of multiple methods. For instance, a combination of WCentroid and Agreement may produce a fusion method that boosts the rankings of relevant links, at the same time respecting the original rankings up to a degree.

References

[1] J. P. Callan, Z. Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks . 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 21–28, Seattle, Washington, 1995. ACM Press.

[2] 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.

[3] 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.

[4] J. Kim, D. Oard, and K. Romanik. Using implicit feedback for user modeling in internet and intranet searching. Technical Report CS-TR-4136, University of Maryland Department of Computer Science Department, 2000.

[5] 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.

[6] Mario Gomez Susan Gauch, Guijun Wang. Profusion: Intelligent fusion from multiple, distributed search engines. volume 2, pages 637–649, 1996.

[7] Geoffrey Towell, Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies for information retrieval. In *Proc. 12th International Conference on Machine Learning*, pages 540–548. Morgan Kaufmann, 1995.

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