A Meta-search Method Reinforced by Cluster Descriptors

Yipeng Shen Dik Lun Lee
Department of Computer Science
The Hong Kong University of Science and Technology
Hong Kong, China
{yipeng, dlee}@cs.ust.hk

Abstract

A meta-search engine acts as an agent for the participant search engines. It receives queries from users and redirects them to one or more of the participant search engines for processing. A meta-search engine incorporating many participant search engines is better than a single global search engine in terms of the number of pages indexed and the freshness of the indexes. The meta-search engine stores descriptive data (i.e., descriptors) about the index maintained by each participant search engine so that it can estimate the relevance of each search engine when a query is received. The ability for the meta-search engine to select the most relevant search engines determines the quality of the final result. To facilitate the selection process, the document space covered by each search engine must be described not only concisely but also precisely. Existing methods tend to focus on the conciseness of the descriptors by keeping a descriptor for a search engine's entire index. This paper proposes to cluster a search engine's document space into clusters and keep a descriptor for each cluster. We show that cluster descriptors can provide a finer and more accurate representation of the document space, and hence enable the meta-search engine to improve the selection of relevant search engines. Two cluster-based search engine selection scenarios (i.e., independent and high-correlation) are discussed in this paper. Experiments verify that the cluster-based search engine selection can effectively identify the most relevant search engines and improve the quality of the search results consequently.

1 Introduction

The enormous and rapidly increasing amount of information on the Web [10] has made the Internet very valuable for discovering useful information. Search engines help users to unearth relevant web pages from the ocean of the Web, which contains billions of documents nowadays. Since the Web is huge and the web pages are updated frequently, the index maintained by a search engine has to be refreshed periodically. This is extremely time and resource consuming because the search engine needs to crawl the Web and download web pages to refresh its index [2]. Thus, it is very difficult for a single search engine to cover a large part, let alone the entirety, of the Web and keep its index up-to-date at the same time.

A search engine indexing a smaller part of the Web may refresh its index more frequently since it focuses on either a specific network domain (e.g., edu.hk) or web servers in a local geographical area. Therefore, a small but focused search engine can return more pertinent and up-to-date web pages compared to a single global search engine. However, its coverage of the Web is small and it may only contain some of the pages relevant to a user query. In order not to miss relevant documents, the user has to visit several search engines individually.

A meta-search engine is a middleware (or agent) consisting of a number of participant search engines [13, 6]. Each participant search engine maintains its own index and offers some descriptive information (i.e., descriptors) about its index to the meta-search engine. The meta-search engine records the descriptors and uses them to estimate the relevance of the underlying search engines when a query is received. To reduce resource consumption, only the most relevant search engines are invoked to process the query. The selection of the most relevant search engines is referred to as the server selection (or ranking) procedure.[1] A good server selection method can effectively identify the most relevant search servers and consequently the most relevant

[1] Since a search engine is often run on a server (often as a server process), we use the terms *search engine* and *search server* interchangeably.

set of documents.

Compared to a single search engine, meta-search engines can provide a larger coverage of the Web and return more up-to-date web pages because it can incorporate a large number of search engines, each of which can index a small part of the Web efficiently. However, the document space covered by each search engine must be described not only concisely but also precisely in the meta-search engine. In general, we want to keep in the meta-search engine more descriptive information about each server's index to make server selection more accurate. However, the large amount of descriptive information requires more resources on the participant search engines to collect and on the meta-search engine to store and process. It is clear that there is a tradeoff between the amount of descriptive data stored at the meta-search engine and the accuracy of the relevance estimation.

Existing server selection methods tend to focus on the conciseness of the descriptors by keeping a descriptor for a search engine's entire index. For example, the descriptive information usually consists of the total number of documents indexed by the search engine, and for each index term the total term frequency, and document frequency, etc [17, 11]. It is clear that information such as "the word *business* appears in 1,000 documents" doesn't mean much without knowing the context within which the word appears.

In this paper, we adopt an approach that clusters a search engine's document space into clusters and represents a document space in the meta-search engine by its clusters descriptors. We propose a new server ranking method which uses the cluster descriptors to estimate a server's relevance. Since clustering can identify homogeneous groups (clusters) of documents in the search engine's document space, cluster descriptors not only give us a finer representation of the document space because each cluster represents only a part of the document space, but also a more accurate representation since words appearing in the same cluster are likely to bear the same meanings. As a result, the selection of relevant search engines will be improved.

It can be argued that clustering is expensive. However, many fast clustering methods have been proposed in the literature [12, 18] to obtain document clusters and their descriptors (e.g., centroids) efficiently. We also show in this paper that it is not necessary to generate a large number of clusters in order to improve the performance and that the descriptors are very concise.

It is important to note that the participant search engines don't have to physically cluster their document spaces. They can retain their own physical organizations and retrieval methods. They only need to implement a clustering module to obtain the cluster descriptors and send them to the meta-search engine. In this sense, the meta-search engine is a *cooperative* system. In fact, all meta-search engines that make use of information provided by the participant search engines, including the existing works described in the next section, are cooperative systems. We believe the economic incentive of getting high visibility and high ranking on a meta-search engine will encourage the participant search engines to provide the descriptive information. Furthermore, although this work has been motivated by web-based meta-search systems, the proposed method can be applied to non-web-based systems which distribute and search documents on different machines for system efficiency, availability, or management reasons.

In this paper, we first define the information kept in a cluster descriptor and then develop the server selection method based on cluster descriptors. Using a document collection derived from TREC[2], we evaluate the performance of the proposed method and compare it to the existing methods. We show that our method has very good performance especially when we only select a few candidates out of a large set of servers for evaluating the query, which is exactly the situation we expect to encounter in a large-scale meta-search engine system.

In the next section, we review the related work. In Section 3, our server ranking method using cluster descriptors is introduced; two ranking scenarios (i.e., independent and high-correlation) are discussed. Section 4 presents some experimental results. The conclusion is drawn in Section 5.

2 Related Work

Meta-search is also referred to as the hierarchical distributed search. Many papers addressing this issue are published recently. As mentioned before, server ranking is the most critical problem in meta-search.

1. *gGloss* server ranking [5, 6] is often regarded as the baseline for other server ranking methods. For each document containing some query terms in a server's index database, it computes the similarity between the document and the query. In *gGloss* ranking, the *ideal*(0) goodness score of the server is the sum of these similarity values.

2. *CORI* [3, 11] adopts the inference network to estimate the relevance of each underlying server. Each query term's document frequencies (*df*s) in all the

[2] *http://trec.nist.gov*

servers and the inverse collection frequency (icf) are used in server ranking, which is basically a $df \times icf$ strategy.

3. *CVV* [17] prefers terms that can distinguish one server from another. Terms with larger variances in the *cue validity* values, which stand for the degree by which one server can differ from others, are given greater weights in the server ranking. Intuitively, larger variance indicates that some more relevant servers can be separated from the irrelevant servers.

4. A recent server ranking method is proposed in [13, 15, 16]. It tries to rank the servers according to the document with the greatest similarity to the query in each server so that the server ranking can guarantee that the most similar documents are always retrieved. For each server, the document with the greatest similarity to the query is supposed to be a document with the maximum normalized term weight for one of the query terms and the average normalized term weights for other query terms. We refer to this server ranking method as *MSD* (the Most Similar Documents).

5. A theoretical model proposed in [4] aims at retrieving a maximum number of relevant documents at the minimum cost. Each server has its own performance curve in terms of recall and precision and there are server-specific costs for the retrieval of documents. Given a query, if the total cost of retrieving n documents from the servers related to the query is minimized, the cost differentials of these servers' cost functions are equal by using Lagrange multipliers, that is, the optimal number of documents to be selected from each of these servers can be anticipated. Moreover, this model can also be extended to optimize the retrieval cost if the user wants to browse n relevant documents.

6. Xu and Croft [14] proposed a cluster-based server ranking method and showed that it is very effective. Given a query, the Kullback-Leibler divergence [14] between each cluster and the query is calculated for each server. The server ranking is based on the most similar cluster in each server. Our method is also cluster based, but we use a different method to estimate a server's relevance from the cluster descriptors.

3 Server Selection

To improve the accuracy of server ranking, we must delineate the document space indexed by a search engine in detail. More data beyond the contents in the present descriptors such as the document frequencies used in *CORI* should be kept at the meta-search engine. Our motivation is to cluster the documents in each underlying server's index database; similar documents are grouped together. The descriptors of the clusters, which are concise in data size and informative in describing the document space, are stored at the meta-search engine. The server selection is based on these descriptors of clusters. In case the documents in a server's index database are not clustered or are already tightly clustered, we may simply treat the documents as one single cluster in the cluster-based server ranking.

Besides, we assume that both the documents in the index databases and the user queries are represented as vectors (i.e., the vector space model is adopted). For each document vector corresponding to a document, the weight of a term in the vector is defined as the raw term frequency of the term divided by the maximum raw term frequency among all the terms in the document. The similarity between any two vectors is the *cosine* value between them, which is the two vectors' *dot product* divided by their two corresponding norms. If $x = (x_1, ..., x_m)$ is a vector, the norm of x is usually defined as $(x_1^2 + ... + x_m^2)^{\frac{1}{2}}$. When we cluster the documents, the similarity between two document vectors is computed. Similarly, when we answer a user query, the similarity between the query vector and a document vector representing the document to be retrieved is computed.

3.1 K-means Clustering

Efficient clustering algorithms have been discussed in [12, 18, 7]. The clustering procedure in these algorithms is based on main memory with the help of some data structures (e.g., the *Clustering Feature Tree* in *BIRCH* [18]). Since these algorithms incur fewer disk operations, their efficiency is significantly improved.

Our objective is to study the effectiveness of the cluster-based server selection method and to determine if an efficient, but not necessarily the most effective, clustering algorithm can still improve the quality of server selection. As such, we adopt the widely used *K-means* clustering algorithm [8], which is an iterative algorithm that converges quickly. The K-means clustering algorithm starts with K rough clusters and iteratively adjusts the clusters until the clusters are satisfactorily refined. Xu and Croft adopted this method and showed that it is quite effective, although the version they used is a more straightforward and faster two-pass (iteration) algorithm [14].

3.2 Descriptors of Clusters

Once the clustering is done, the documents in the same cluster can be briefly represented by that cluster's descriptor, which contains some basic information about the cluster. Each cluster c's descriptor consists of the following data:

- The total number of documents in the cluster, N.
- The centroid of the cluster, $(aw_1, aw_2, ..., aw_m)$. aw_i is the average term weight of term t_i over all the documents in the cluster including the documents do not contain t_i.
- The document frequency vector, $(df_1, df_2, ..., df_m)$. df_i is the number of documents that contain t_i in the cluster.

For term t_i, aw_i is the average value among all the documents no matter whether the documents contain the term or not. Given a query with multiple terms, only the documents containing the query terms are related to the query. Thus, another vector $(pw_1, pw_2, ..., pw_m)$, which is referred to as the purified weight vector, can be deduced. $pw_i = aw_i \cdot N/df_i$ is the average term weight of term t_i among all the documents containing term t_i in the cluster.

3.3 Similarity Weights of Clusters

Our server ranking method is based on the cluster descriptors. For each query, we calculate the similarity of a server according to the similarity weights of its underlying clusters. Assume $q = (qt_1, qt_2, ..., qt_m)$ is the query vector; $p_i = df_i/N$ is the probability that t_i is contained in a random document in the cluster. sim_c, the similarity of cluster c to the query, can be estimated via two different scenarios, namely, the *independent scenario* and the *high-correlation scenario.*

3.3.1 Independent Scenario

The independent scenario assumes that the distribution of terms in the documents is mutually independent. For example, suppose $q = (t_1, t_2)$ and p_1 and p_2 are the probabilities that a document contains t_1 and t_2 respectively. The probability that the document contains both t_1 and t_2 is $p_1 \cdot p_2$ due to the independent distribution assumption. p_r, the probability that a random document in the cluster contains either t_1 or t_2, or both t_1 and t_2, is equal to $p_1 + p_2 - p_1p_2 = 1 - (1 - p_1)(1 - p_2)$. If q contains m terms, p_r, the probability that a random document contains some query terms, is defined as:

$$p_r = 1 - (1 - p_1)(1 - p_2)...(1 - p_m).$$

Thus, $p_r \cdot N$ is the total number of documents in the cluster containing some query terms. To simplify the computation, we choose the purified weight vector as the representative of these documents. If sim_r is the similarity between the purified weight vector and the query vector, sim_c, the estimated similarity of cluster c, is defined as sim_r times the number of documents that contain some query terms in c:

$$\begin{aligned} sim_c &= p_r N sim_r & (1) \\ &= [\ 1 - (1 - p_1)(1 - p_2)...(1 - p_m)\ ]\ N\ sim_r. \end{aligned}$$

3.3.2 High-Correlation Scenario

Our high-correlation scenario is more or less similar to that of *gGloss* [5] despite that our specification is slightly different. We make two assumptions in the same manner as those made in *gGloss* [5]:

1. Assume $p_1 \leq p_2$. If a document contains t_1, then it will also contain t_2.
2. The weight of a term is distributed uniformly over all the documents containing the term.

For example, if $q = (t_1, t_2, t_3)$ and $p_1 \leq p_2 \leq p_3$, the probability that a random document contains all of the three terms is p_1, which corresponds to the representative vector $rv_1 = (pw_1, pw_2, pw_3)$; the probability that a document exactly contains t_2 and t_3 is $p_2 - p_1$, which corresponds to the representative vector $rv_2 = (0, pw_2, pw_3)$, and so on.

Generally, if q has m terms $t_1, t_2, ..., t_m$ and $p_1 \leq p_2 \leq ... \leq p_m$, then $ph_i = p_i - p_{i-1}$ $(i > 1)$ is the probability that a document exactly contains $t_i, t_{i+1}, ..., t_m$, which corresponds to the representative vector $rv_i = (0, ..., 0, pw_i, pw_{i+1}, ..., pw_m)$. ph_1, the probability that a document contains all m terms is equal to p_1, which corresponds to $rv_1 = (pw_1, ..., pw_m)$.

Assume sim_{r_i} is the similarity between vector rv_i and query q. sim_c, the estimated similarity between cluster c and q can be computed as follows:

$$\begin{aligned} sim_c &= \sum_{i=1}^{m} ph_i\ N\ sim_{r_i} & (2) \\ ph_i &= \begin{cases} p_1, & i = 1, \\ p_i - p_{i-1}, & i > 1. \end{cases} \end{aligned}$$

Furthermore, if sim_{r_i} is defined as the *dot product* between the query vector and the corresponding representative vector rv_i, and the *inverse document frequencies* are incorporated in the representative vector, then sim_c is equal to the ideal *gGloss* weight (i.e., the $ideal(0)$ weight for cluster c).

3.4 Estimated Weights of Servers

For each query, we may use one of the aforementioned scenarios to estimate the similarity of a cluster. Suppose the documents indexed by a search server s can be clustered into k clusters $c_1, c_2, ..., c_k$. We define the estimated similarity of s as the sum of the cluster weights of the top clusters. That is, assume that sim_{c_i} is the estimated similarity weight of c_i and that $sim_{c_1} \geq sim_{c_2} \geq ... \geq sim_{c_k}$. Then, sim_s, the estimated similarity of s, is defined as:

$$sim_s = \sum_{i=1}^{l} sim_{c_i} \ , \ 1 \leq l \leq k \quad . \tag{3}$$

In our server weight formula, the top clusters are supposed to contain the majority of the relevant documents within s's index. Therefore, the clusters with smaller estimated similarity weights can be ignored in estimating s's similarity. Furthermore, if the number of clusters in s is large, this method can reduce the amount of data stored at the meta-search engine, which selects the most similar servers when a query is received.

4 Experimental Results

To justify the effectiveness of our cluster-based server ranking method, a simulation environment consisting of 50 pseudo-servers is constructed. Documents from the Volume 4 and Volume 5 of the TREC collection, consisting of over 500,000 documents of total size of about 2.1 Gbytes, are distributed randomly (uniformly) among the 50 servers. If a skew distribution is adopted, given a query, the difference of the relevance of the underlying servers tends to be more obvious and larger servers are often very likely to be more relevant. The experiments are based on the uniform distribution of documents in this paper since it is more difficult to effectively identify the most relevant servers when the documents are uniformly distributed. The documents are indexed by each server and are further clustered. The cluster descriptors are kept at the meta-search engine, where the servers' similarity values are estimated against each user query.

For each query, the meta-search engine only selects the most similar servers for processing the query. The number of selected servers is referred to as the *cast number*. Each selected server evaluates the query and returns the retrieved documents together with their corresponding *TFIDF* values; the documents are sorted in descending order of the *TFIDF* values [1].

The local *inverse document frequencies* are used in the *TFIDF* formula since it has been shown that the effects of *global inverse document frequencies* and local inverse document frequencies on the ranking are about the same especially when the documents are uniformly distributed in the servers [13]. Furthermore, since the web queries are usually short (about two terms on average), both the effectiveness and the necessity of adopting the inverse document frequencies are weakened. More importantly, maintaining global information is impractical when the meta-search engine consists of a large number of participant search engines.

Finally, the meta-search engine merges the documents retrieved from the selected servers to a final result list and returns it to the user.

The fifty queries used in the experiments are the TREC topics 301-350; the short query format is adopted since the web queries are usually short [9]. Each short TREC query contains 2.48 terms on average. Furthermore, since the number of returned documents is usually large for the web queries and the user often just browses a small number of them, the precision of the top documents is taken as the most basic performance criterion in our experiments.

4.1 Number of Clusters

If we generate a large number of clusters from the documents covered by a search server, each cluster tends to contain a smaller number of highly similar documents. Thus, the cluster descriptors would be more precise in describing the documents in the clusters. However, this will produce more cluster descriptors, which will take up more storage space at the meta-search engine. Furthermore, the cost of generating a large number of clusters is high.

In this experiment, we will examine the effect on retrieval precision when different numbers of clusters are used. We use the similarity weights of all the clusters in a server's index to generate the similarity weight of the server. The precision values are obtained when the top 10, 20, and 30 documents are considered. From Figure 1, we notice that the precision values initially increase when the number of clusters increases. However, once the total number of clusters exceeds a certain value (about 12 clusters in the experiment), increasing it further will not affect the precision values much. This is an important result, since it means that we don't have to spend a lot of resources, in terms of both cluster generation and descriptor storage costs, to generate a large number of clusters in order for our method to work well.

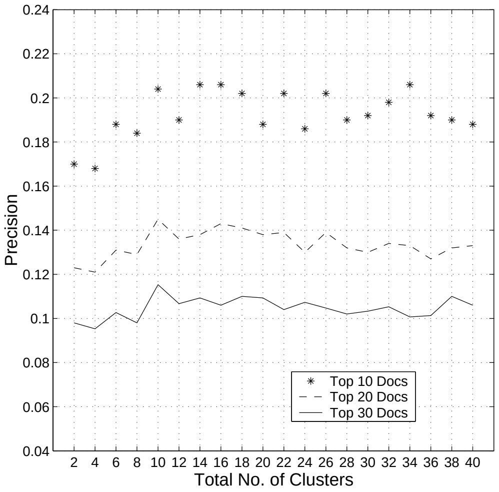


Figure 1. Effect of the Total Number of Clusters (Cast Number=5, Independent Scenario)

4.2 Evaluation Ratio

In our cluster-based server selection method, we propose to sum up the similarity values of its top clusters when the similarity weight of a server is estimated. The percentage of the clusters used in calculating the weight of a server is referred to as the *evaluation ratio*. If the evaluation ratio equals 1, the similarity weights of all the clusters in the server are used. The rationale is that the relevant documents of a query should be similar to each other and as such should be contained in a small number of clusters. These clusters in turn should have high cluster similarity weights to the query. Therefore, the clusters with the highest similarity weights should contain most of the relevant documents. We try to verify this intuition with an experiment. The result is shown in Figure 2. We can see that when we use more than 30% top clusters (i.e., the evaluation ratio>0.3) to compute the server weight, the precision of the top documents does not improve very much. More clusters employed in the ranking only result in some fluctuation in the performance. This is also verified in Figure 3, where the two ranking scenarios are compared.

4.3 Ranking Scenarios

When we estimate the similarity of a cluster in a server, we have considered the independent and high-correlation ranking scenarios. For the queries employed in this experiment, we examine the performance of our server ranking method under both scenarios. The result is depicted in Figure 3, which shows that the independent ranking scenario performs better than the high-correlation scenario for various evaluation ratios. We believe that this phenomenon is attributed to the fact that the query terms employed in the TREC topics are not highly correlated. Therefore, we hereafter adopt the independent ranking scenario for our cluster-based server selection method.

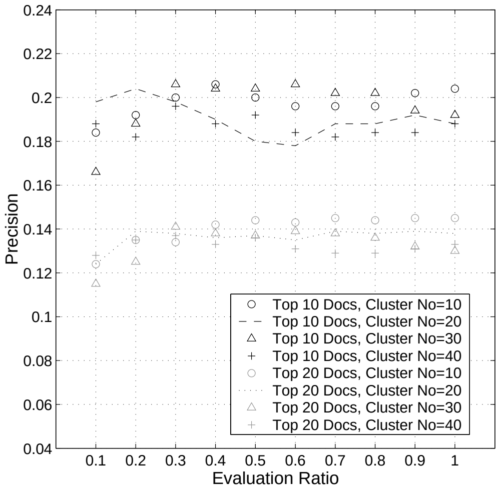


Figure 2. Effect of the Evaluation Ratio (Cast Number=5, Independent Scenario)

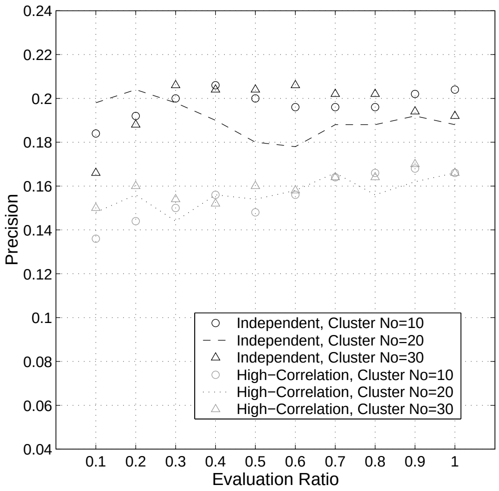


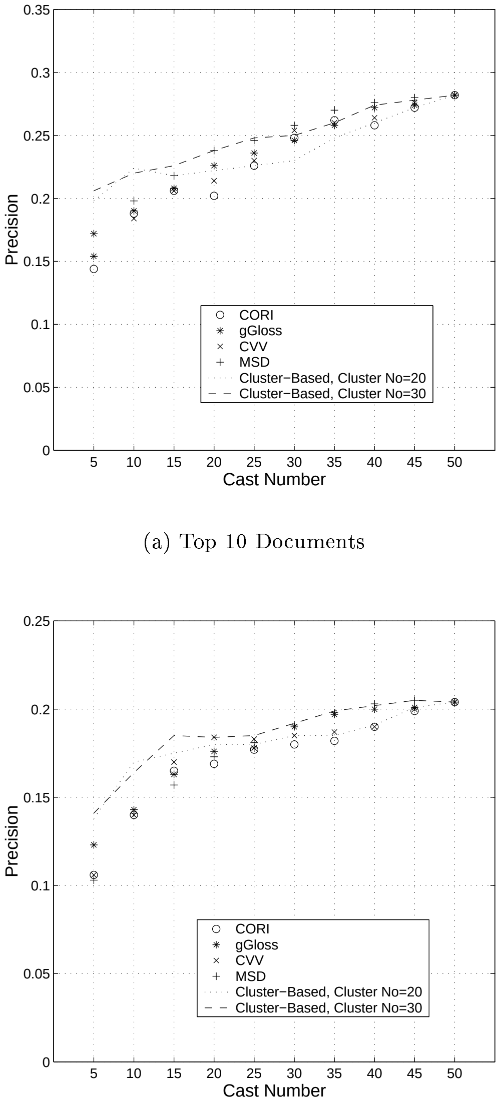
Figure 3. Ranking Scenarios (Cast Number=5)

4.4 Comparison

In this experiment, we compare our cluster-based server selection method against four existing methods described in Section 2. We evaluate our method when the number of clusters generated from the documents under each search server is 20 or 30. The evaluation ratio for both cases is 0.3. We also examine the result of the ranking methods when the precision values of the top 10 and top 20 documents are compared.

The result of the experiment is shown in Figure 4. When the cast number is small (e.g., cast number=10), we can see that our method is significantly better than the other ranking methods under the two cluster numbers and the two top document numbers. When the cast number increases, the differences between the methods diminish. It is obvious that when all of the servers are selected (i.e., cast number=50), the five methods become the same.

Since the number of servers in a meta-search engine is expected to be very large, we can only select a small number of relevant servers to evaluate a query in order to speed up query execution and save system resource. Our cluster-based method can identify the most relevant servers accurately even if the cast number is small. Therefore, it is very suitable for the large-scale meta-search systems.



(a) Top 10 Documents

(b) Top 20 Documents

Figure 4. Different Server Ranking Methods

5 Conclusion

Clustering is a useful method to identify data patterns. We suggest in this paper that the document space covered by a search server can be clearly and concisely represented if it is clustered and represented by cluster descriptors. Consequently, the relevance of a search engine to a user query can be estimated more accurately. We show by experiments that our cluster-based server selection is very effective in identifying the relevant search servers for a query. This is very important for a large-scale meta-search engine since the server selection mechanism must be able to identify a small number of relevant servers from a large number of underlying servers.

The experiments also reveal some basic aspects of the cluster-based server selection. To characterize the document space with satisfactory accuracy, the documents in the index do not have to be clustered into a large number of clusters. Instead, a moderate number of clusters is sufficient to describe the document space. This reduces the clustering cost as well as the storage space at the meta-search engine. Furthermore, it is enough to use a small percentage of the cluster descriptors in estimating a server's relevance to the query. Thus, for each term, it is only necessary to store the information of the most similar clusters; this results in reduced storage cost with guaranteed performance.

Finally, since only the cluster descriptors are used in ranking the servers, the underlying server does not need to cluster the documents physically. It only needs to periodically update the cluster descriptors and send them to the meta-search engine. As a result, faster or incremental clustering algorithms aiming at producing rough clusters can be designed in the same manner as the clustering feature tree used in *BIRCH* [18].

6 Acknowledgment

The writing of this chapter was supported by Research Grants Council of Hong Kong, China (Project numbers HKUST-6154/98E).

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Essex, England, 1999.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *WWW7 / Computer Networks*, 30(1-7):107–117, 1998.

[3] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, pages 21–28, 1995.

[4] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[5] L. Gravano and H. Garcia-Molina. Generalizing GLOSS to vector-space databases and broker hierarchies. In *VLDB*, pages 78–89, 1995.

[6] L. Gravano, H. Garcia-Molina, and A. Tomasic. GLOSS: Text-source discovery over the Internet. *ACM Transactions on Database Systems (TODS)*, 24(2):229–264, 1999.

[7] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *SIGMOD*, pages 73–84, 1998.

[8] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[9] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the Web. *SIGIR Forum*, 32(1):5–17, 1998.

[10] S. Lawrence and C. L. Giles. Accessibility of information on the Web. *Nature*, Vol. 400:107–109, July 1999.

[11] A. L. Powell, J. C. French, J. P. Callan, and M. Connell. The impact of database selection on distributed searching. In *SIGIR*, pages 232–239, 2000.

[12] C. Silverstein and J. O. Pedersen. Almost-constant-time clustering of arbitrary corpus subsets. In *SIGIR*, pages 60–66, 1997.

[13] Z. Wu, W. Meng, C. T. Yu, and Z. Li. Towards a highly-scalable and effective metasearch engine. In *International World Wide Web Conference (WWW10)*, 2001.

[14] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *SIGIR*, pages 254–261, 1999.

[15] C. T. Yu, W. Meng, K.-L. Liu, W. Wu, and N. Rishe. Efficient and effective metasearch for a large number of text databases. In *CIKM*, pages 217–224, 1999.

[16] C. T. Yu, W. Meng, W. Wu, and K.-L. Liu. Efficient and effective metasearch for text databases incorporating linkages among documents. In *SIGMOD*, 2001.

[17] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the Internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–50, Melbourne, Australia, 1997.

[18] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *SIGMOD*, pages 103–114, 1996.