A Meta-Search Method with Clustering and Term Correlation

Dyce Jing Zhao   Dik Lun Lee   Qiong Luo

Department of Computer Science
Hong Kong University of Science & Technology
`{zhaojing,dlee,luo}@cs.ust.hk`

Abstract. A meta-search engine propagates user queries to its participant search engines following a server selection strategy. To facilitate server selection, the meta-search engine must keep concise descriptors about the document collections indexed by the participant search engines. Most existing approaches record in the descriptors information about what terms appear in a document collection, but they skip information about which documents a keyword appears in. This results in ineffective server ranking for multi-term queries, because a document collection may contain all of the query terms but not all of the terms appear in the same document.
In this paper, we propose a server ranking approach in which each search engine's document collection is divided into clusters by indexed terms. Furthermore, we keep the term correlation information in a cluster descriptor as a concise method to estimate the degree of term co-occurrence in a document set. We empirically show that combining clustering and term correlation analysis significantly improves search precision and that our approach effectively identifies the most relevant servers even with a naïve clustering method and a small number of clusters.

1 Introduction

Search engines, such as Google, are efficient tools for users to find useful information on the Web by simple keyword-based queries. With such a vast amount of data on the Web and the rapid pace at which the Web is growing, it is a daunting (if not impossible) task for a single search server to index the entire Web and to keep its index repository up-to-date. As an alternative to a single search engine, a meta-search engine, or meta-server, accepts queries from users, propagates the queries to one or more of its participant search servers, and returns to users a uniform representation of the query results retrieved by those servers. This meta-search method achieves a better coverage of the Web with less update costs than a centralized search method, even though each participant search engine covers only a specific topic or a small domain of the Web.

Server selection or server ranking is a crucial procedure in a meta-search system. When a user query is made, the server selection procedure identifies a subset of the participant search engines that are potentially most relevant to the query, in order to promptly obtain query results of good quality. As a result, server selection is the first step in determining the efficiency and effectiveness of a meta-search method. In this paper, we explore how to improve server selection in order to enhance the effectiveness of a meta-search, especially for long queries.

Most of the existing server selection methods [11] keep statistical information about the participant search servers' indexes. They invariably maintain a list of terms found in the indexed documents, while the weight of a term may vary across different methods. When a multi-term query arrives, the meta-search engine computes a relevance score of each server for each single query term. The scores of each server are then summed to produce the final rank of the server. Apparently, terms in a query are considered in isolation from one another during this process. Therefore, whether two query terms appear in one document or each appears in a different document is not taken into consideration. Consequently, such methods in general are ineffective for multi-term queries.

Considering term correlation for multi-term queries is necessary for effective server selection. However, the computation of term correlation information may be time-consuming and the accuracy of the results may be interfered with by noisy data. This leads us to consider clustering in combination with term correlation analysis in server ranking. Specifically, we propose to divide each search engine's document collection into clusters based on their term occurrences and then to compute the term correlation information within each cluster. This can be done efficiently and effectively, because a cluster is smaller than the entire collection and contains similar documents. The resulting server rank strategy considers not only the matching level of individual query terms at a server, but also that of term correlation at the server.

In addition to designing and implementing our server ranking strategy with a combination of clustering and term correlation, we have performed simulation experiments on a subset of the TREC (Text Retrieval Conference) documents collection (details are given in Section 5). Our empirical results show that this method works well in estimating search server relevance. Most notably, even though clustering is often a computationally expensive task, our method of combining clustering and term correlation analysis yields good results with a naïve but fast clustering algorithm and a small number of clusters.

The remainder of the paper is organized as follows. Section 2 briefly discusses some related work. Section 3 presents the clustering and term correlation methods we used. Section 4 describes our server ranking method with clustering and term correlation. We give the experimental results in Section 5, and draw some conclusions in Section 6.

2 Related Work

A great number of meta-search systems have been developed in recent years, together with various server selection methods. In addition, there have been proposals on distributed search [10][13]. In the following we discuss a few previous meta-search methods, some of which consider document clustering or term correlation in server ranking and therefore are especially relevant to our work.

SavvySearch[8] adjusts the number of concurrent participant search engines to the network load and the local CPU load at the query time, and ranks the participant search servers by relevance score (TFIDF), user click-through and their recent performance.

Clustering has been applied in a meta-search method [12], in which descriptors of clusters instead of the entire indexed data collections are adopted. The descriptors are computed at the individual search servers and are kept by the meta-search server. This results in improved scalability with respect to computational consumption and storage space.

Document relevance has also been considered in a previous meta-search engine [15]. This meta-search engine computes the relevance score of an individual search server S to a term T as the similarity between T and the most relevant document in S, and records only the top N relevant servers. It offers excellent scalability and guarantees that the K most relevant documents are retrieved for single-term queries. However, it doesn't perform well with multi-term queries and tends to return answers pointing to identical sources.

Finally, the Ingrid[10] search infrastructure provides a good model for clustering based on term combinations rather than separate terms, but it is not likely to scale up, since the number of clusters required to maintain a correct Ingrid topology increases sharply as the system grows.

3 Clustering and Term Correlation

3.1 Clustering Algorithm

Clustering has been studied extensively in machine learning [6], pattern recognition [5], and optimization [3]. The most general approach to clustering is to view it as a density estimation problem [14]. Various models [9] have been developed for clustering purposes, among which the K-means algorithm is well-known for its efficiency.

K-means is a widely-used iterative algorithm that starts from K rough centroids and converges gradually. Much effort has been made to improve its effectiveness [2]. Since an effective clustering algorithm is often expensive and thus is unsuitable for server ranking, we purposely adopt a naïve but fast two-iteration K-means algorithm in our server ranking method. We show by experiments that our method works well, notwithstanding the relatively high error rate of the clustering algorithm.

3.2 Term Correlation

Descriptors of Clusters After the document collections of the participant search servers are partitioned into clusters, we can obtain crucial information that best describes each cluster. We define the descriptor of a cluster c as follows:

- The cardinality of cluster c, $Card(c)$;
- The centroid of c, $Cent(c) = (aw_1, aw_2, ..., aw_n)$, where each coordinate value aw_i is the average of all weights of term t_i across the documents in cluster c, with the weight of t_i in a document being zero if there is no occurrence of this term in the document.
- A term correlation matrix, $CM(c)$, which records the pairwise similarity of frequent terms of cluster c. Similarity and frequent terms are defined in Sections 3.2.2 and 3.2.3.

Since clustering is based on the density distribution of the indexed documents in the vector space, the resulting clusters are composed of documents that are similar to each other to some extent. We can regard a cluster as a special topic area in the sense that documents of the same topic tend to be relatively similar. While a clustering algorithm may make mistakes due to noise in the data and may be too expensive to be applied to the entire document collection of a search server, it is more efficient and effective to analyze term correlation within specific topic areas, where the term space is narrowed and the noisy data are eliminated.

Frequent Term Selection If it is assumed that clusters are topic areas, the frequent terms of a cluster can then be regarded as popular words of a special topic. We choose the frequent term set, $FT(c)$, for a cluster c, such that $FT(c)$ covers the majority of popular words of c, but does not have much intersection with the frequent term sets of the other clusters.

For each document vector $dv = (tw_1, tw_2, ..., tw_n)$ in a cluster c, tw_i is the weight of term t_i in the document, which is defined as the raw term frequency tf_i divided by the maximum raw term frequency tf_{max} in dv. In this paper, we define the total weight of a term t_i in cluster c as follows:

$$W(t_i, c) = \sum_{dv \in c} tw_i, \; dv = (tw_1, tw_2, ..., tw_n).$$

We compare a threshold with the total weight of each term in the cluster to determine the frequent term set of the cluster. This method identifies frequent terms based on their absolute total weight values within a cluster, regardless of the cluster size, and therefore tends to produce a larger frequent term set for a larger cluster, which is natural in the real world.

Term Correlation Representation After the frequent term set is decided on, we analyze the correlation among frequent terms within a cluster by computing a term correlation matrix, through which information about the distribution of terms over documents can be obtained.

To compute the term correlation matrix, $CM(c)$, of a cluster c, we first construct a term-document matrix, $DM(c)$, from c's document collection. An entry $dm_{i,j}$ in $DM(c)$ is the weight of term t_i in document vector $dv_j \in c$. Thus, we obtain a term-document vector for each frequent term.

We define the correlation of two terms t_i and t_j as the similarity between their term-document vectors, tv_i and tv_j. Let $Card(c) = m, tv_i = (w_{i,1}, w_{i,2}, ..., w_{i,m})$ and $tv_j = (w_{j,1}, w_{j,2}, ..., w_{j,m})$. $cm_{i,j}$, the (i, j) entry of the term correlation matrix, $CM(c)$, can be represented as follows:

$$\begin{aligned} cm_{i,j} &= sim(tv_i, tv_j) \\ &= \frac{\sum_{k=1}^{m} w_{i,k} \times w_{j,k}}{\sqrt{\sum_{k=1}^{m} w_{i,k}^2 \times \sum_{k=1}^{m} w_{j,k}^2}}. \end{aligned} \tag{1}$$

Equation 1 implies that terms with a similar distribution of occurrences among the documents within a cluster have a high correlation value. Specifically, when two terms have the exact same distribution (i.e., they co-occur in all documents with the same weight), their term correlation is one; if they do not co-exist in any single document, their term correlation is zero.

4 Server Ranking

Having presented the cluster descriptors with term correlation information, we are now ready to discuss our server ranking method in a multi-term query meta-search.

Consider a query with two terms A and B. If the cluster descriptors fail to distinguish between one document with terms A and B, and two documents each with only one of the terms, the referral may be false. Our goal in server ranking is to eliminate such false referrals and subsequent fruitless query results by ranking servers based on the term correlation within clusters.

We adopt a cluster ranking metric based on the cardinality, the centroid, and the term correlation matrix of each cluster. We then rank each search server by summing the weights of all of its clusters. This way, we guarantee that a cluster with query terms co-occurring more frequently is assigned to a higher rank and that a server with higher quality clusters is also ranked higher.

Let us define the ranking metric formally as follows. Suppose a search server $S(c_1, c_2, ..., c_k)$ has K Clusters, and that we define the rank of the server as the sum of the cluster ranks (weights).

$$W(S) = \sum_{i=1}^{K} W(c_i).$$

Consider a query Q with k query terms. We represent query Q as a vector qv where all terms that occur in the query have a weight of 1, and 0 otherwise. Then, the weight of a cluster c with respect to query Q is represented as follows, where sim is a function of the cosine similarity between two vectors as defined in Equation 1.

$$W(c) = \{\alpha \times sim[Cent(c), qv] + \beta \times (\sum_{i<j} cm_{i,j})\} \times Card[c]; \tag{2}$$

$$for\ all\ (i,j) \in \{(i,j)|\ i < j, t_i \in Q, t_j \in Q\}.$$

In this cluster weight formula, α and β are two weighting coefficients decided at the meta-search server side, where

$$\alpha + \beta = 1.$$

This equation takes into account the proximity between the cluster centroid and the query vector, the correlation of the query terms within the cluster, and the size of the cluster. It shows that the closer the cluster is located to the query vector, the more often the query terms co-exist in the documents of the cluster; and the larger the cluster, the higher rank the cluster is assigned.

Let us illustrate this by a simple example. Consider a query with four terms A, B, C, and D, and two clusters c_1 and c_2. Cluster c_1 has two documents, document doc_{11} containing query terms A, B, and C simultaneously, and document doc_{12} containing query term D only. Cluster c_2 also has two documents, document doc_{21} containing all of the four query terms, and document doc_{22} containing none of the query terms. Intuitively, cluster c_2 should be ranked higher than c_1.

Let us see how our method works in this example. To ensure that all other conditions are equal, we assume

- $Cent(c_1) = Cent(c_2)$, and that the similarity between the query vector and the cluster centroid is S.
- $Card(c_1) = Card(c_2) = 2$ (i.e., no other documents are contained in the two clusters besides $doc_{11}, doc_{12}, doc_{21}$, and doc_{22}).
- The weights of all query terms, A, B, C, and D, have a value of 1 in whichever documents they occur.

Since doc_{11} and doc_{12} in cluster c_1 together contribute to $sim(A, B)$, $sim(B, C)$, and $sim(A, C)$, while in c_2, only doc_{21} contributes to the similarity of all the $C_4^2 = 6$ possible term pairs, we compute the cluster weights to be

$$W(c_1) = (\alpha * S + \beta * 3) * 2; \quad W(c_2) = (\alpha * S + \beta * 6) * 2.$$

This shows that cluster c_2 is indeed ranked higher than c_1 by our method.

We believe that our server selection method is a scalable approach, because the growth of the number of frequent terms in a cluster is slow with respect to that of the number of documents in the cluster. It follows that the computation cost of the term correlation matrix is linear to the number of documents when the vocabulary is stabilized and the size of the matrix is invariable.

Finally, we believe that document updates (which often occur on the Web) can easily be handled by our approach if we periodically perform clustering and update the cluster descriptors. Since all the computation of clustering and term correlation analysis is done off-line and is distributed over all participant search servers, the update overhead will not be too heavy a burden at each local server.

5 Experimental Results

To evaluate our server selection method, we set up a simulation environment in which a large document set is partitioned into smaller collections, each of which is indexed and searched separately. Thus, each collection simulates a "pseudo"-search server. In our experiment, 50 pseudo-search servers are used. The document set comes from Volumes 4[1] and 5[2]of the TREC collection, consisting of over 500,000 documents. We distribute the data among all 50 pseudo-servers such that all servers maintain document collections of a similar cardinality. We choose a uniform distribution in this paper, rather than a skewed one (i.e., having a bias towards some of the servers over others with respect to the number of indexed documents), to isolate the effects of our method from other factors.

Each pseudo-server is partitioned into K clusters using the two-iteration K-means clustering algorithm with random initial points. All cluster descriptors are computed at individual search servers and are stored at the meta-search engine. For each query, the meta-search engine performs the server selection and propagates the query to the top T highest ranked search servers. We choose retrieval accuracy (precision) as our search performance measurement metric, as it is the most important metric in Web search, where users are mostly interested in only the top few query results (precision or accuracy, not recall).

Given a query Q, the cast number T (i.e., the number of search servers to which a user query is forwarded) and the number of top relevant documents retrieved at each search engine, N, the retrieval accuracy of the meta-search engine is defined as:

$$Acc(Q,T,N) = \frac{|(\cup_{i=1}^{S} R_{l_i} \cap R|}{N},\ for\ all\ s_{l_i}\ selected,$$

where R_i is the set of query results returned by search server s_i, and R is the set of top N relevant documents across the entire TREC document collection. We believe this measurement describes the effectiveness of a server ranking approach well, since it is decided mostly by which servers are selected rather than by the quality of the document collection.

5.1 Coefficient Settings

In the cluster weight formula (formula 2), the two coefficients α and β respectively suggest how much the ranking score is affected by the statistical information and the term correlation within clusters. We conduct experiments on various α and β settings in an environment where the number of clusters $K = 20$, and the cast number $N = 20$.

Figure 1 illustrates the retrieval precision of three typical settings, from which we can observe that $\alpha = 0.2,\ \beta = 0.8$ works the best.

[1] TREC Volume 4, May 1996 Collection includes material from the Financial Times (1991, 1992, 1993, 1994), the Congressional Record of the 103rd Congress (1993), and the Federal Register (1994).

[2] TREC Volume 5, April 1997 Collection includes material from the Foreign Broadcast Information Service (1996) and the Los Angeles Times (1989, 1990).

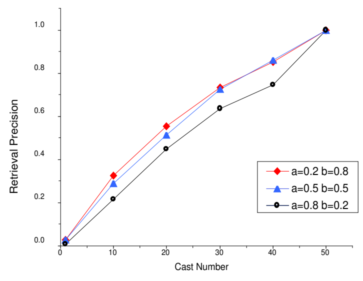

Fig. 1. Various α and β settings

5.2 Comparison

We then evaluate our approach for $K = 20; 30,\ N = 10; 20$, and $T = 1; 10; 20; 30; 40; 50$. It is obvious that when $T = 50$, the retrieval accuracy of all approaches converges to 100%. We compare our method only with the method proposed by Shen et al. [12], which makes use of clustering but not term correlation analysis, because their method has been shown to perform better than the other existing ranking methods such as gGloss[7], CORI[4], and CVV[16].

We refer to our method TC and that of Shen et al. CL .[3] in the figures. Obviously, the two methods perform the same for single-term queries. Therefore, we consider only multi-term queries in our experiments. The results (Figures 2-5) show that our approach (TC) consistently achieves a higher accuracy than the cluster-based method (CL).

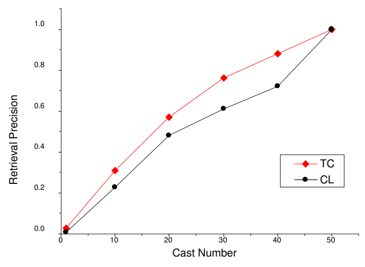

Fig. 2. 20 Clusters, Top 10 Documents

Our results also show that $K = 30$ is a good value for clustering in our setting and that our approach improves retrieval accuracy more significantly

[3] TC: Clustering and term correlation-based method
CL: Cluster-based method without term correlation analysis

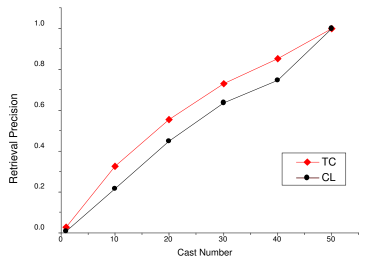

Fig. 3. 20 Clusters, Top 20 Documents

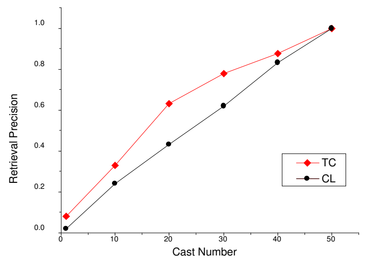

Fig. 4. 30 Clusters, Top 10 Documents

when a small number of top-score documents are retrieved ($N = 10$), which is especially encouraging since it is most important to precisely retrieve the very top document set.

Finally, we conduct experiments where the number of clusters is rather small ($K = 5$) and thus the clustering procedure is quite fast. In this scenario, the cluster-based method with no term correlation analysis turns out to be as ineffective as a random server selection method. In comparison, we show in Figure 6 that our cluster and term correlation-based method still achieves a high accuracy.

6 Conclusion

In this paper, we have proposed a server ranking method combining document clustering and term correlation for meta-search systems with respect to multi-term queries. In our method, each participant search server applies clustering to its local repository and computes the correlation matrix of frequent terms to be included in the cluster descriptor. This approach takes into consideration the degree of co-occurrence of query terms in documents. Therefore, it significantly improves the effectiveness of server selection for multi-term queries. This is confirmed by our initial experiments on the TREC document collection. Additionally, our empirical results show that a simple but fast clustering algorithm

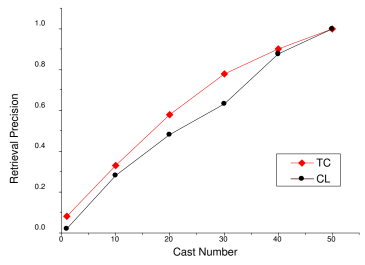

Fig. 5. 30 Clusters, Top 20 Documents

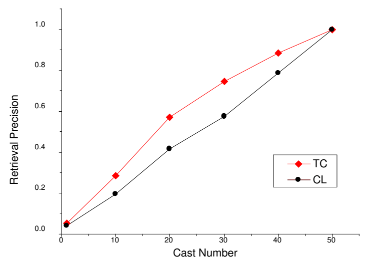

Fig. 6. 5 Clusters, Top 20 Documents

and a small number of clusters is sufficient to achieve the accuracy enhancement, thus allaying concerns about the overhead of the clustering procedure.

7 Acknowledgements

This work was supported by a grant from the Research Grant Council of Hong Kong, China (Grant No. HKUST 6154/98E, HKUST6079/01E, DAG02/03.EG03). We would like to thank Yunpeng Li for his helpful comments.

References

1. Bergman, M.XK., *The Deep Web: Surfacing Hidden Value*, BrightPlanet Company, LLC, Jul 2000.
2. Bradley, P.S., Fayyad, U.M., *Refining Initial Points for K-Means Clustering*, Proc. 15th International Conference on Machine Learning (ICML), pages 91-99, Madison, Wisconsin, USA, Jul 1998.
3. Bradley, P.S., Mangasarian, O.L., Street, W.N., *Clustering via Concave Minimization*, Advances in Neural Information Processing Systems 9 (NIPS), pages 368-374, Dec 1996.

4. Callan, J.P., Lu, Z., Croft, W.B., *Searching Distributed Collections with Inference Networks*, Proc. 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 21-28, Washington, DC, USA, Jul 1995.
5. Duba, R.O., Hart, P.E., *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.
6. Fisher, D., *Knowledge Acquisition via Incremental Conceptual Clustering*, Machine Learning, 2:139-172, 1987.
7. Gravano, L., Garcia-Molina, H., Tomasic, A., *GLOSS: Text-source Discovery over the Internet*, ACM Transactions on Database Systems (TODs), vol. 24, pages 229-264, Jun 1999.
8. Howe, A., Dreilinger, D., *A MetaSearch Engine that Learns Which Search Engines to Query*, AI Magazine, 18(2), 1997.
9. Meila, M., Heckerman, D., *An Experimental Comparison of Several Clustering and Initialization Methods*, Technical Report MSR-TR-98-06, Microsoft Research, Redmond, WA, Feb 1998.
10. Francis, P., Kambayashi, T., Sato, S., Shimizu, S., *Ingrid: A Self-Configuring Information Navigation Infrastructure*, Proc. 4th International World Wide Web Conference (WWW), pages 519-537, Dec 1995.
11. Powell, A.L., French, J.C., Callan, J.P., Connell, M., *The Impact of Database Selection on Distributed Searching*, Proc. 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 232-239, Athens, Greece, Jul 2000.
12. Shen, Y., Lee, D.L., *A Meta-Search Method Reinforced by Descriptors of Clusters*, Proc. 2nd International Conference on Web Information Systems Engineering (WISE), pages 129-136, Kyoto, Japan, Dec 2001.
13. Shen, Y., Lee, D.L., *An MDP-based Peer-to-Peer Search Server Network*, Proc. 3rd International Conference on Web Information Systems Engineering (WISE), pages 269-278, Singapore, Dec 2002.
14. Silverman, B.W., *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, London, 1986.
15. Wu, Z., Meng, W., Yu, C.T., Li, Z., *Towards a Highly-scalable and Effective Metasearch Engine*, Proc. 10th International World Wide Web Conference (WWW), pages 386-395, Hong Kong, China, May 2001.
16. Yuwono, B., Lee, D.L., *Server Ranking for Distributed Text Retrieval Systems on the Internet.* Proc. 5th International Conference on Database Systems for Advanced Applications (DASFAA), pages 41-50, Melbourne, Australia, Apr 1997.