A Hierarchic Architecture for Conceptual Information Retrieval

Shih-Hao Li and Peter B. Danzig
Computer Science Department
University of Southern California
Los Angeles, California 90089-0781
{shih, danzig}@cs.usc.edu

Abstract

Conceptual retrieval returns information related to a specific topic but not restricted to a query term. A common approach is to compare the query with all the documents in the database. When the number of documents is large, the searching time becomes significant. In this paper, we propose a hierarchic architecture which integrates latent semantic indexing (LSI) and hierarchic agglomerative clustering to reduce the searching time. We employ three clustering algorithms (single link, complete link, and group average) and conduct experiments on four standard document collections (CACM, CISI, CRAN, and MED). The experimental results show our method requires less searching time while maintaining comparable retrieval effectiveness as non-clustered search.

1 Introduction

Searching information by conceptual meanings often suffers from the vocabulary problem [1], which states that users may fail to obtain desired information if the query terms used are different from those indexed by the retrieval system. To address the vocabulary problem, Deerwester et al. proposed Latent Semantic Indexing (LSI) [2], where queries and documents are represented as vectors of conceptual meanings. LSI compares a query with all the documents in the database, then returns those with higher similarity to the user. It has been tested on several information systems with promising results [2].

A deficiency of LSI is that it searches through the whole database. The searching time becomes significant when the database is large. One way to ameliorate this problem is to search documents by clusters instead of each individual record. This approach is based on van Rijsbergen’s cluster hypothesis [3], where he stated closely associated documents tend to be relevant to the same queries. He also proposed cluster-based retrieval on hierarchically clustered collections to improve retrieval effectiveness and efficiency [3, 4]. In this paper, we describe a hierarchic architecture that applies hierarchic clustering to LSI documents, and compare its performance with non-clustered LSI retrieval on several document collections. We review the methodology of LSI and hierarchic clustering in Section 2. We describe our architecture in Section 3 and show the experimental results in Section 4. Section 5 presents the conclusions.
2 Background

2.1 Latent Semantic Indexing

LSI is an extension of Salton's Vector Space Model [5], in which documents and queries are represented as vectors of term frequencies or weights. To capture the semantic structure among documents in a database, LSI applies Singular Value Decomposition (SVD) to a term-document matrix representing the database and generates vectors of \( k \) (typically 100 to 300) orthogonal indexing dimensions, where each dimension represents a linearly independent concept. The \( k \)-dimensional vectors are used to represent both documents and terms in the same semantic space, while their values indicate the degrees of association with the \( k \) underlying concepts. Figure 1 shows SVD applies to a term-document matrix.

![Diagram of SVD process](image)

Figure 1: SVD applies to an \( m \times n \) term-document matrix, where \( m \) and \( n \) are the numbers of terms and documents in the database, and \( k \) is the indexing dimension used by SVD.

A query vector in LSI is the weighted sum of its component term vectors. For example, a \( p \)-term query is represented as the average sum of the \( p \) decomposed term vectors. To determine relevant documents, the query vector is compared with all the document vectors, and those with the highest cosine coefficient [6] are returned. Because the indexing dimension \( k \) is chosen much smaller than the number of terms and documents in the database (i.e. the number of rows and columns in the term-document matrix), those \( k \) concepts are neither term nor document frequencies but are compressed forms of both. Therefore, a query can hit documents without having common terms but with common concepts.

2.2 Hierarchic Agglomerative Clustering

To cluster LSI documents, we apply the hierarchic agglomerative clustering method. Hierarchic agglomerative clustering has been studied to increase retrieval effectiveness and efficiency as compared to the conventional search of non-clustered data [4, 7, 8, 9, 10].

A typical hierarchic clustering method can be described as follows:

1. Compute all pair-wise inter-document similarity coefficients.
2. Place each document in a cluster of its own.
3. Form a new cluster by merging the most similar pair of current clusters.
4. Recompute the similarity between the newly merged cluster and the remaining clusters.
5. Repeat step 3 while there is more than one cluster.

The output of a hierarchical clustering algorithm is a cluster hierarchy as shown in Figure 2. Various clustering methods differ in the manner in which they define the similarity between clusters. Three of the most commonly used methods in information retrieval are single link, complete link, and group average. Single link clustering uses the similarity between the most similar pair of documents (one in each cluster) as the similarity between the two clusters. Complete link clustering uses that of the least similar document pair in the two clusters. Group average clustering uses the average similarity of all document pairs to be the inter-cluster similarity. Early experiments showed that the performance of hierarchical clustering methods varies when tested on different document collections. [4, 9, 11, 10].

![Figure 2: Sample cluster hierarchy, where documents are denoted as squares and the similarities between merged clusters are shown in circles.](image)

3 System Architecture

We integrate LSI with hierarchical agglomerative clustering to improve the efficiency of conceptual retrieval. As shown in Figure 3, we build a hierarchical system by applying one of the above clustering methods to arrange documents at different levels.

For a $N$-level system, we choose $N$ similarity thresholds, one for each level. At the $i^{th}$ level, the clusters with similarity larger than or equal to the $i^{th}$ lowest threshold are grouped together. Each of the remaining clusters forms a group of its own. Thus, the whole database can be represented as a set of clusters at a specific level. Note that a higher level clusters can be further divided into sub-clusters at a lower level. The cluster hierarchy of a database is created only once. Figure 4 shows a 3-level system hierarchy generated from the cluster hierarchy in Figure 2.

To search relevant documents, a query is compared with each cluster centroid, defined as the average vector of all the documents in the cluster. To keep the volume of query results acceptable, we set a cutoff ($C$) for the desired number of returned documents. In the following, we describe the
Figure 3: System Architecture.

Figure 4: A 3-level system hierarchy with similarity thresholds 0.3, 0.4, and 0.6 generated from the cluster hierarchy in Figure 2. Clusters are labeled with their size shown in brackets.
steps for searching and returning relevant documents. First, the cluster centroids at the top level (i.e. level \( N \)) are compared with each query and ranked in descending order of their similarities with the query. The required number of documents are then taken from the top of the ranking. If the first-ranked cluster has less than \( C \) documents, all the documents are returned. The next cluster in the list is checked until a sufficient number of documents are returned. If a cluster has more than \( C \) documents, the search moves downward one level. The same procedure is performed at this level. This process continues until it reaches the lowest cluster level (i.e. level 1), where clusters contain no sub-clusters but documents. Based on the cluster hypothesis, documents in the same cluster are equally similar to the same query. We randomly select documents from the last of the retrieved clusters until sufficient documents are obtained.

Below, we use Figure 4 as an example, where \( C \) is set to 3. At the top level, we assume cluster \( b \) has higher similarity with the query than cluster \( a \). We return all the documents in cluster \( b \) because it has less than \( C \) documents. Then we check cluster \( a \). Since it has more than \( C \) documents, we compare all its child clusters with the query. Assume clusters \( d \) and \( h \) have higher similarity than clusters \( c \) and \( g \), respectively. We randomly select one of the documents in cluster \( h \) as the last document satisfying \( C \).

4 Experiments

To evaluate our method, we conduct experiments on four standard document collections (CACM, CISI, CRAN, and MED), for which queries and relevant judgments are available. We apply all the three hierarchical clustering methods (single link, complete link, and group average) to generate the cluster hierarchy and compare their retrieval performance on each collection.

4.1 Methodology

Documents in each database are indexed with terms occurring in the title and abstract but not on a stop list of 429 common words. While queries are written in natural language, terms in a query are used only if they do not appear on the same stop list and if they appear in at least one document. All indexed terms are stored in their original forms without stemming. Table 1 shows the characteristics of each collection.

<table>
<thead>
<tr>
<th></th>
<th>CACM</th>
<th>CISI</th>
<th>CRAN</th>
<th>MED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of documents</td>
<td>3204</td>
<td>1460</td>
<td>1400</td>
<td>1033</td>
</tr>
<tr>
<td>Number of queries</td>
<td>64</td>
<td>112</td>
<td>225</td>
<td>30</td>
</tr>
<tr>
<td>Mean number of terms per document</td>
<td>23.93</td>
<td>48.83</td>
<td>57.85</td>
<td>57.20</td>
</tr>
<tr>
<td>Mean number of terms per query</td>
<td>10.61</td>
<td>30.03</td>
<td>9.19</td>
<td>10.13</td>
</tr>
<tr>
<td>Mean number of relevant documents per query</td>
<td>12.44</td>
<td>27.80</td>
<td>8.17</td>
<td>23.20</td>
</tr>
</tbody>
</table>

Table 1: Collection characteristics.

Each document is represented by a vector calculated by LSI with indexing dimension 100. We compute the cosine similarity between each pair of documents and apply the three hierarchical clustering methods. The implementations of those clustering methods are based on the algorithms developed by Voorhees [12]. In our experiments, we use a two-level architecture. The upper level
consists of clusters generated using similarity threshold 0.50, while the lower level contains clusters with similarity threshold 0.75. Table 2 shows the number of clusters and their average size generated in the two-level architecture using different clustering methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>CACM</th>
<th>CISI</th>
<th>CRAN</th>
<th>MED</th>
</tr>
</thead>
<tbody>
<tr>
<td>single</td>
<td>1496 (2.14)</td>
<td>1373 (1.06)</td>
<td>1260 (1.11)</td>
<td>831 (1.24)</td>
</tr>
<tr>
<td>link</td>
<td>96 (33.38)</td>
<td>327 (4.46)</td>
<td>289 (4.84)</td>
<td>94 (10.99)</td>
</tr>
<tr>
<td>complete</td>
<td>1945 (1.65)</td>
<td>1386 (1.05)</td>
<td>1297 (1.08)</td>
<td>899 (1.15)</td>
</tr>
<tr>
<td>link</td>
<td>1070 (2.99)</td>
<td>914 (1.60)</td>
<td>837 (1.67)</td>
<td>501 (2.06)</td>
</tr>
<tr>
<td>group</td>
<td>1775 (1.81)</td>
<td>1383 (1.06)</td>
<td>1282 (1.09)</td>
<td>877 (1.18)</td>
</tr>
<tr>
<td>average</td>
<td>96 (33.38)</td>
<td>748 (1.95)</td>
<td>637 (2.20)</td>
<td>345 (2.99)</td>
</tr>
</tbody>
</table>

Table 2: Number of clusters and their average size (in parentheses) generated in a two-level architecture.

To evaluate the retrieval performance of our experiments, we calculate the effectiveness measure, $E$, defined as [3]:

$$E = 1 - \frac{(1 + \beta^2)PR}{\beta P + R},$$

where $P$ and $R$ are the precision and recall values, and $\beta$ is a parameter reflecting the relative importance of recall to precision defined by the user. For example, $\beta = 2.0$ indicates recall is twice as important as precision. The $E$ measure has been commonly used in the studies of document clustering. It varies from 0 to 1, in which low values correspond to high retrieval effectiveness. In addition to $E$ measure, we also measure the efficiency of our methods by counting the number of comparisons between the query and the cluster centroids to obtain sufficient documents. We compare the results of using the three clustering methods with our standard, full search, which matches queries with each document in the database, ranks them in descending order of their similarity with the query, and returns only the top cutoff documents.

### 4.2 Results

Figures 5, 6, and 7 show the average $E$ values with $\beta = 0.5, 1.0$, and 2.0 over all the queries on the four databases using a cutoff of 20, 50, and 100 documents, respectively. We can see full search has, in general, the lowest $E$ values comparing to the other three clustering methods. This is because full search returns the most similar documents selected from the whole database instead of a partial number of clusters. Next to full search, complete link has the second best overall performance, followed by group average and single link. Because complete link tends to generate a large number of small and tightly bound clusters, most of the documents in a cluster might be relevant to a query if their cluster centroid has a high similarity with the query. In contrast to complete link, single link usually forms a small number of large clusters with little internal cohesion. Therefore, it is likely to cause low precision and results in poor retrieval effectiveness. Group average is a compromise between single link and complete link. It generates clusters with medium size.

Figure 8 shows the average number of comparisons as the percentage of full search for the four databases using different methods. In Figure 8, we can see group average is the most efficient method in terms of number of comparisons to obtain sufficient documents. It requires comparisons.
Figure 5: The average $E$ values using a cutoff of 20 documents for full search, complete link, group average, and single link on the four databases (a) CACM, (b) CISI, (c) CRAN, and (d) MED, respectively.
Figure 6: The average $E$ values using a cutoff of 50 documents for full search, complete link, group average, and single link on the four databases (a) CACM, (b) CISI, (c) CRAN, and (d) MED, respectively.
Figure 7: The average $E$ values using a cutoff of 100 documents for full search, complete link, group average, and single link on the four databases (a) CACM, (b) CISI, (c) CRAN, and (d) MED, respectively.
as few as 17.9% of that by full search. Complete link is a close second, which ranges from 33.4% to 62.7% comparing to full search. Single link requires as many as 89.7% in the worst case. In contrast to the retrieval effectiveness, the efficiency is improved significantly when using cluster-based search. Table 3 shows the percentage of average improvement on retrieval effectiveness and efficiency of the three clustering methods against full search.

<table>
<thead>
<tr>
<th>Method</th>
<th>CACM</th>
<th>CACM</th>
<th>CRAN</th>
<th>CRAN</th>
<th>MED</th>
<th>MED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E(%)$</td>
<td>$F(%)$</td>
<td>$E(%)$</td>
<td>$F(%)$</td>
<td>$E(%)$</td>
<td>$F(%)$</td>
</tr>
<tr>
<td>single link</td>
<td>-0.15</td>
<td>55.95</td>
<td>-0.13</td>
<td>16.08</td>
<td>-0.04</td>
<td>31.02</td>
</tr>
<tr>
<td>complete link</td>
<td>-0.07</td>
<td>66.56</td>
<td>-0.21</td>
<td>37.34</td>
<td>-0.05</td>
<td>40.15</td>
</tr>
<tr>
<td>group average</td>
<td>-0.08</td>
<td>82.19</td>
<td>-0.16</td>
<td>48.63</td>
<td>-0.06</td>
<td>54.24</td>
</tr>
</tbody>
</table>

Table 3: Percentage of improvement on retrieval effectiveness ($E$) and efficiency ($F$) using the three clustering methods against full search. The negative values represent a loss of performance.

As shown in Table 3, the improvements of retrieval efficiency are significantly larger than the loss of retrieval effectiveness when using cluster-based search. The results show that clustering methods improve retrieval efficiency from 16% to 82% while still providing over 99% retrieval effectiveness of that by full search.

5 Conclusions

We have proposed a hierarchic architecture to speedup the searching time for conceptual information retrieval. Our method employs hierarchic clustering on documents indexed by latent semantic indexing. The experimental results show the improvements of retrieval efficiency using cluster-based search outperforms non-clustered search significantly, while the loss of retrieval effectiveness is less than 1%. This indicates our method can provide much faster query response in conceptual information retrieval while maintaining equivalent retrieval effectiveness as the conventional method.

References


Figure 8: The average number of comparisons as percentage of full search using a cutoff of (a) 20, (b) 50, and (c) 100 documents for full search, complete link, group average, and single link on the four databases.


