7.1 Efficient scoring and ranking

7.1.6 Cluster pruning

In cluster pruning we have a preprocessing step during which we cluster the document vectors. Then at query time, we consider only documents in a small number of clusters as candidates for which we compute cosine scores. Specifically, the preprocessing step is as follows:

1. Pick $\sqrt{N}$ documents at random from the collection. Call these leaders.
2. For each document that is not a leader, we compute its nearest leader.

We refer to documents that are not leaders as followers. Intuitively, in the partition of the followers induced by the use of $\sqrt{N}$ randomly chosen leaders, the expected number of followers for each leader is $\approx N / \sqrt{N} = \sqrt{N}$. Next, query processing proceeds as follows:

1. Given a query $q$, find the leader $L$ that is closest to $q$. This entails computing cosine similarities from $q$ to each of the $\sqrt{N}$ leaders.
2. The candidate set $A$ consists of $L$ together with its followers. We compute the cosine scores for all documents in this candidate set.

The use of randomly chosen leaders for clustering is fast and likely to reflect the distribution of the document vectors in the vector space: a region of the vector space that is dense in documents is likely to produce multiple leaders and thus a finer partition into sub-regions. This illustrated in Figure 7.3.

Variations of cluster pruning introduce additional parameters $b_1$ and $b_2$, both of which are positive integers. In the pre-processing step we attach each follower to its $b_1$ closest leaders, rather than a single closest leader. At query time we consider the $b_2$ leaders closest to the query $q$. Clearly, the basic scheme above corresponds to the case $b_1 = b_2 = 1$. Further, increasing $b_1$ or
Figure 7.3
Figure 7.3  Cluster pruning.

Exercise 7.1
We suggested above (Figure 7.2) that the postings for static quality ordering be in decreasing order of $g(d)$. Why do we use the decreasing rather than the increasing order?

Exercise 7.2
When discussing champion lists, we simply used the $r$ documents with the largest $tf$ values to create the champion list for $t$. But when considering global champion lists, we used $idf$ as well, identifying documents with the largest values of $g(d) + tf-idf$. Why do we differentiate between these two cases?

Exercise 7.3
If we were to only have one-term queries, explain why the use of global champion lists with $r = K$ suffices for identifying the $K$ highest scoring documents. What is a simple modification to this idea if we were to only have $s$-term queries for any fixed integer $s > 1$?

Exercise 7.4
Explain how the common global ordering by $g(d)$ values in all high and low lists helps make the score computation efficient.