Due: 2018/04/08

Homework 5

Problem 1. (20 points) We suggested Figure 1 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?

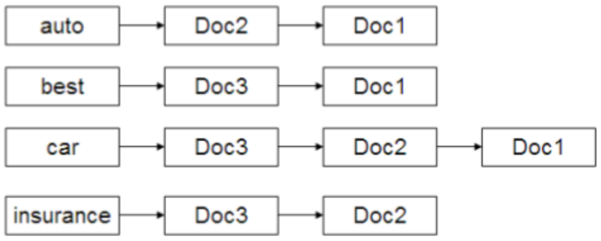

Figure 1: A static quality-ordered index. In this example we assume that Doc1, Doc2 and Doc3 respectively have static quality scores $g(1) = 0.25$, $g(2) = 0.5$, $g(3) = 1$.

Solution:
Here we only required that all the postings be ordered by a single common ordering. We rely on the $g(d)$ to provide this common ordering. The first idea is a direct extension of champion lists: for a well-chosen value r we maintain for each term t a global champion list of the r documents with the highest values for $g(d) + tf - idf_{t,d}$ The list itself is, like all the postings lists considered so far, sorted by a common order (either by document IDs or by static quality). Then at query time, we only compute the net scores for documents in the union of these global champion lists. Intuitively, this has the effect of focusing on documents likely to have large net scores.
We conclude the discussion of global champion lists with one further idea. We maintain for each term t two postings lists consisting of disjoint sets of documents, each sorted by $g(d)$ values. The first list, which we call high, contains the m documents with the highest tf values for t. The second list, which we call low, contains all other documents containing t. When processing a query, we first scan only the high lists of the query terms, computing net scores for any document on the high lists of all (or more than a certain number of) query terms.If we obtain scores for K documents in the process, we terminate. If not, we continue the scanning into the low lists, scoring documents in these postings lists.
Since documents with higher $g(d)$ have higher scores, a decreasing order is good for efficient top-K retrieval.

Problem 2. (30 points) 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 $s > 1$?

Solution:
if we only got the one-term queries then what we will do take the union of the champion lists for each of the terms comprising the query and restrict cosine computation to only the documents in the union set. If the query contains only one term, we just take the list with r = K, because there is no need to compute the union.
As for the each term it identifies the $[\frac{K}{s}]$ As the highest scoring element.
For a one-term query, the ordering of documents is independent of the weight of the term in the query. It only depends on the weight of the term in the documents. The truncated weight-ordered postings list contains the K documents with the highest weights. Thus, it is identical to the ranked list for the one-term query.

Problem 3. (30 points) Explain how the common global ordering by $g(d)$ values in all high and low lists helps make the score computation efficient.

Solution:
In the general case, the postings list of query terms are traversed to find the documents which contains all or most of query terms and then scores are calculated for these documents to find the top K. Thus more importance is given to $v(q)$ $.v(d)$ in the short listing stage and $g(d)$ is accounted for later. If common global ordering by $g(d)$ is implemented, not just it increases the efficiency of results but also less documents would have to undergo score calculation process thereby reducing some work through at the cost of initial ordering

Problem 4. (20 points) Let the static quality scores for Doc1, Doc2 and Doc3 in Figure 2 be respectively 0.25, 0.5 and 1. Sketch the postings for impact ordering when each postings list is ordered by the sum of the static quality score and the Euclidean normalized tf values in Figure 2.

	Doc1	Doc2	Doc3
car	0.88	0.09	0.58
auto	0.10	0.71	0
insurance	0	0.71	0.70
best	0.46	0	0.41

Figure 2: Euclidean normalized tf values for documents.

Solution:
$car- > d3- > d1- > d2$ $auto- > d2 - .D3 - .D1$ $insurance- > d3- > d2- > d1$
$best- > d3- > d1 - .D2$