CS4051
Information Retrieval
Week 05

Muhammad Rafi
February 22, 2023

Computing Scores in a Complete Search System

Chapter No. 7

Agenda

- Efficient Scoring and Ranking
- Component of Retrieval Systems

Efficient Score Ranking

- So far, we only optimize the mathematical implementations on VSM
- Cosine Similarity or Angle between doc and query.
- We really interested in relative scores (rather than exact scores).
- For any document d, the cosine similarity V(q) · v(d) is the weighted sum, over all terms in the query q, of the weights of those terms in d.
- This scheme computes a score for every document in the postings of any of the query terms; the total number of such documents may be considerably smaller than N.

Efficient Score Ranking

```
FASTCOSINESCORE(q)
 float Scores[N] = 0
for each d
do Initialize Length[d] to the length of doc d
for each query term t
do calculate w_{t,q} and fetch postings list for t
    for each pair(d, tf_{t,d}) in postings list
    do add wf_{t,d} to Scores[d]
Read the array Length[d]
for each d
do Divide Scores[d] by Length[d]
return Top K components of Scores[]
```

Efficient Score Ranking

- Inexact Top K documents
  - Our focused is on retrieving precisely the K highest-scoring documents for a query.
  - It suffices to retrieve K documents whose scores are very close to those of the K best.
- Heuristics
  - Find a set A of documents that are contenders, where $K < |A| \ll N$. A does not necessarily contain the K top-scoring documents for the query, but is likely to have many documents with scores near those of the top K.
  - Return the K top-scoring documents in A.

Efficient Score Ranking

- Index Elimination (using additional heuristics)
  - We only consider documents containing terms whose idf exceeds a preset threshold. Thus, in the postings traversal, we only traverse the postings
  - We only consider documents that contain many (and as a special case, all) of the query terms.
- Champion List (Precomputed List)
  - The idea of champion lists (sometimes also called fancy lists or top docs) is to precompute, for each term t in the dictionary, the set of the r documents with the highest weights for t; the value of r is chosen in advance.
  - For tfi*df weighting, these would be the r documents with the highest tf values for term t. We call this set of r documents the champion list for term t.

Efficient Score Ranking

- Champion List (Retrieval)
  - Given a query q we create a set A as follows: we take the union of the champion lists for each of the terms comprising q.
  - We now restrict cosine computation to only the documents in A.
- Static Quality Score
  - In many search engines, we have available a measure of quality g(d)for each document d that is query-independent and thus static.
  - The net score for a document d is some combination of g(d) together with the query-dependent score induced Cosine(q,d).

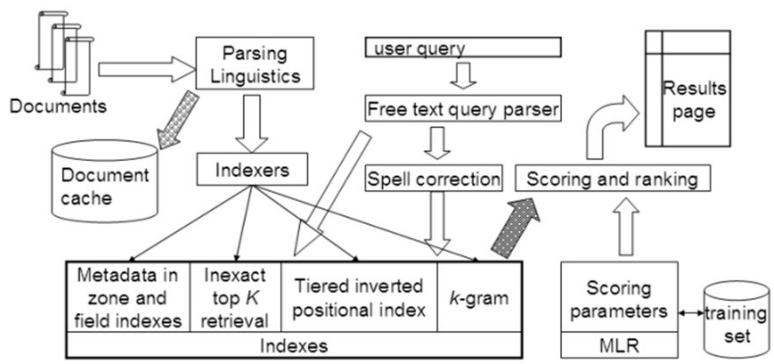
Efficient Score Ranking

- Net-Score and Ordering
  - The net-score for a pair of (q,d) is a sum of scores on g(d) + cosine(d,q).
  - There are two important ideas:
    - Only process posting list with unordered documents with high tf scores for query term. Drop the threshold for only top r documents.
    - consider the query terms in decreasing order of idf, so that the query terms likely to contribute the most to the final scores are considered first.

Efficient Score Ranking

- 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.
  - How to create clusters?
    - Pick $\sqrt{N}$ documents at random from the collection. Call these leaders.
    - For each document that is not a leader, we compute its nearest leader.
  - Query processing proceeds as follows:
    - 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.
    - The candidate set A consists of L together with its followers. We compute the cosine scores for all documents in this candidate set.

Component of Retrieval Systems



► **Figure 7.5** A complete search system. Data paths are shown primarily for a free text query.

Conclusion

- Retrieval Optimization is still an active area of research for large IR systems.
- Most of the big player in IR systems space have their proprietary techniques for optimization.