CS317

Information Retrieval

Week 07

Muhammad Rafi

March 04, 2024

Computing Scores in Complete Search Systems(VSM)

Chapter No. 7

Agenda

- Efficient scoring and ranking
- Inexact Top k documents
- Index Elimination
- Champion Lists
- Static Quality Scores
- Impact Ordering
- Cluster Pruning
- Conclusion

Efficient Scoring and Ranking

- Modified Cosine computation as discussed in the last lecture.
  - Only compute the partial scores for each q term and document d (only common dimensions will add values to score).
  - Should we go for all N document- obviously not.
  - We are only interested in k top ranked documents
  - How we can make it efficient?
    - If we know in advance that which documents are high-scoring for a given query?
    - If we can arrange all the documents score in an efficient top retrieval data structures (Max. Heap)
    - Any rough estimates of ranking which can be computed quickly will be helpful.

Fast Cosine Scores

```
FastCosineScore(q)
 1  float Scores[N] = 0
 2  for each d
 3  do Initialize Length[d] to the length of doc d
 4  for each query term t
 5  do calculate w_{t,q} and fetch postings list for t
 6     for each pair(d, tf_{t,d}) in postings list
 7     do add wf_{t,d} to Scores[d]
 8  Read the array Length[d]
 9  for each d
10  do Divide Scores[d] by Length[d]
11  return Top K components of Scores[]
```

Inexact Serach

- Inexact Top Key Document Retrieval
  - Find a set A of documents that are contenders, where K < |A| ≪ 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.
- Index Elimination
- This is based on two key factors:
  - We only consider documents that contain many (and as a special case, all) of the query terms (tf).
  - Only Consider idf query terms with high value(not highest)

Champion List

- Precompute for each dictionary term t, k documents of highest weight in it's postings.
  - Call it champion list for term t
  - Only compute the scores for k documents.
  - This k may not be same for every term t.
  - Pick the top (10-20 documents based on scores)

Static Quality Scores

- Top ranking documents should be both "Relevant" and "Authoritative"
  - Relevant – Terms / cosine scores
  - Authoritative – document host (PageRank)
- Modeling Authority
  - Assign a query independent score to each document – quality score (0-1) – g(d) { static score}
- NET Score
  - NET-score (q,d) = g(d) + cos(q,d), {weighted combination}
  - Return top K documents on NET-Score {may be k <A}
- Global Champion List

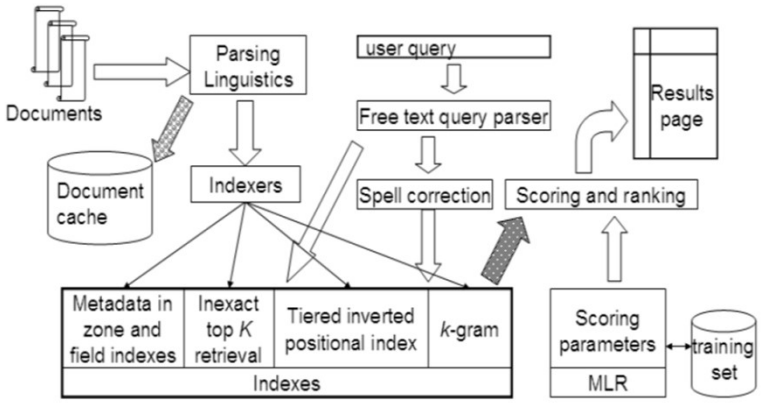
Impact Ordering

- Early Termination
  - Process t posting list with fixed k documents.
  - Stop for documents having wf(t,d) < alpha
  - For each term in query – terminate early and get a union for all selected documents.
- idf – ordering terms
  - High idf contribute more to score
  - Stop if the doc score is relatively unchanged for next documents.

Cluster Pruning

- Preprocessing
  - Pick some landmark documents for each term
  - From all other documents – apply knn to link these documents to a landmark document. (Leader <-> Followers)
- Query Processing
  - Given a q find the nearest leader (landmark document)
  - Get the k nearest document to this leader.
  - Apply Cosine similarity to these

Complete Search Systems



General Strategies – search

- Tiered Index - A common solution to this issue is the user of tiered indexes, which may be viewed as a generalization of champion lists.
- Query-term proximity - Consider a query with two or more query terms, t1 , t2, . . . , tk. Let ω be the width of the smallest window in a document d that contains all the query terms, measured in the number of words in the window.
- Intuitively, the smaller that ω is, the better that d matches the query.
- Proximity weighting is very subjective.

Conclusion

- Large Scale search systems – perform a lot of optimizations
- If we can present results in a Proxy form and we can make a user HAPPY. There is no issue at all.
- She will never feel the difference.