Ch.7 Computing scores in a complete search system

浙江大学微软技术俱乐部
Microsoft Technology Club
Zhejiang University

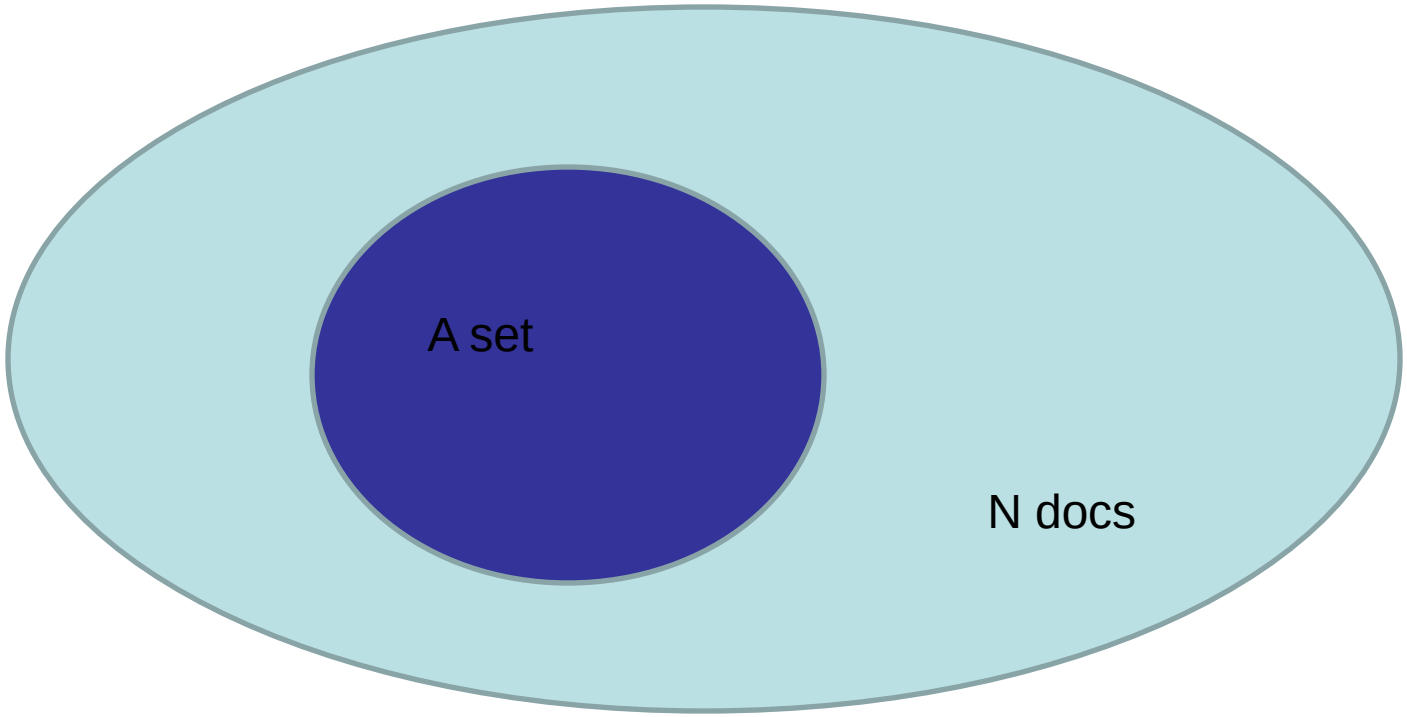
Efficient scoring and ranking


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

Weight of the term are absence, so $w_{t,q}$ is all the same

Inexact top K document retrieval



We use top K of the A set, instead of top K of the N docs.

Inexact top K document retrieval

The key word: Heuristics

1. We only consider document containing terms whose idf exceeds a preset threshold.
2. We only consider documents that contain many of the query terms (<k docs?).

Inexact top K document retrieval

The key word: precompute

1. For each term t in the dictionary, the set of the r documents with the highest weights for t.
2. Given a query q we create a set A as: we take the union of the champion list for each of the terms comprising q.

Inexact top K document retrieval

The key word: static

1. We compute a static quality scores g(d), which is query-independent.
2. Net-score(q, d)=g(d)+similarity(q, d)
3. In posting list, order the posting doc by g(d)
4. A global champion list by term t with the weight g(d)+tf-idf(t,d)
5. We have two list:
    1. A champion list;
    2. A list contains all other documents containing t.

Inexact top K document retrieval

The key word: impact ordering

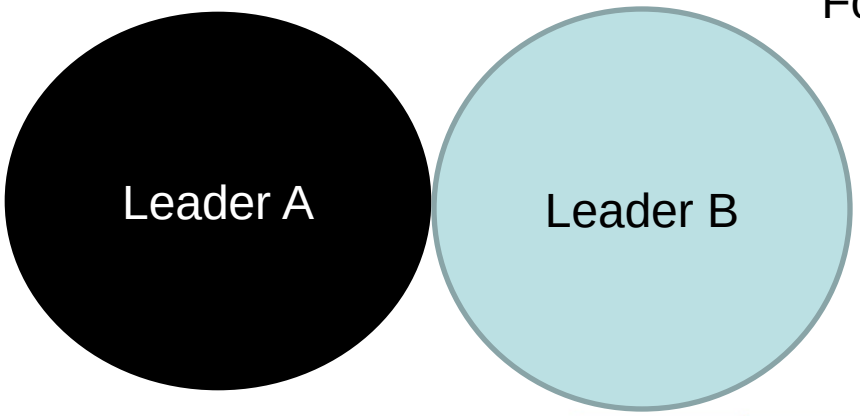
1. Order the documents d in the postings list of term t by decreasing order of $tf_{t,d}$
2. Use this posing list:
    1. When traversing the postings list fro a query term t, we stop after a prefix of the posting list.
    2. When accumulating scores in the outer loop, we consider the query terms in decreasing order of idf.

Inexact top K document retrieval

The key word: cluster

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

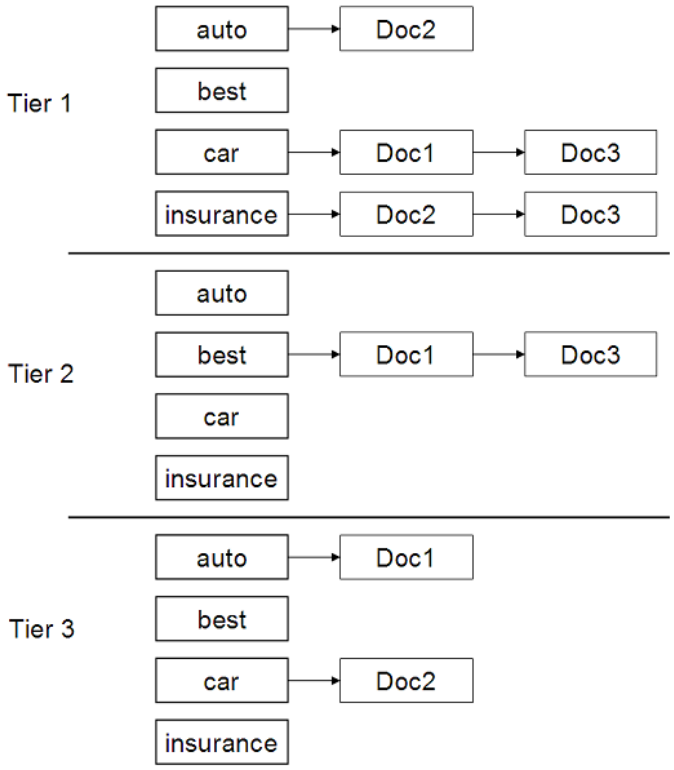
For a query, view its KNN Leaders



Components of an information retrieval system

Tiered Index

If we fail to get k tops in tier N,
We can fall in tier N+1



Query-term proximity

The smallest windows size w which contains all the query terms.

Generate a proximity-weighted scoring function to depend on w.

Designing parsing and scoring functions

Operators query ← Parsing function — Free text query

Document set → Scoring function → Search result list

$$score(d_j, q_j) = g \cdot s_T(d_j, q_j) + (1 - g) s_B(d_j, q_j).$$

$$\varepsilon(g, \Phi_j) = (r(d_j, q_j) - score(d_j, q_j))^2,$$

$$\sum_j \varepsilon(g, \Phi_j).$$

Putting it all together