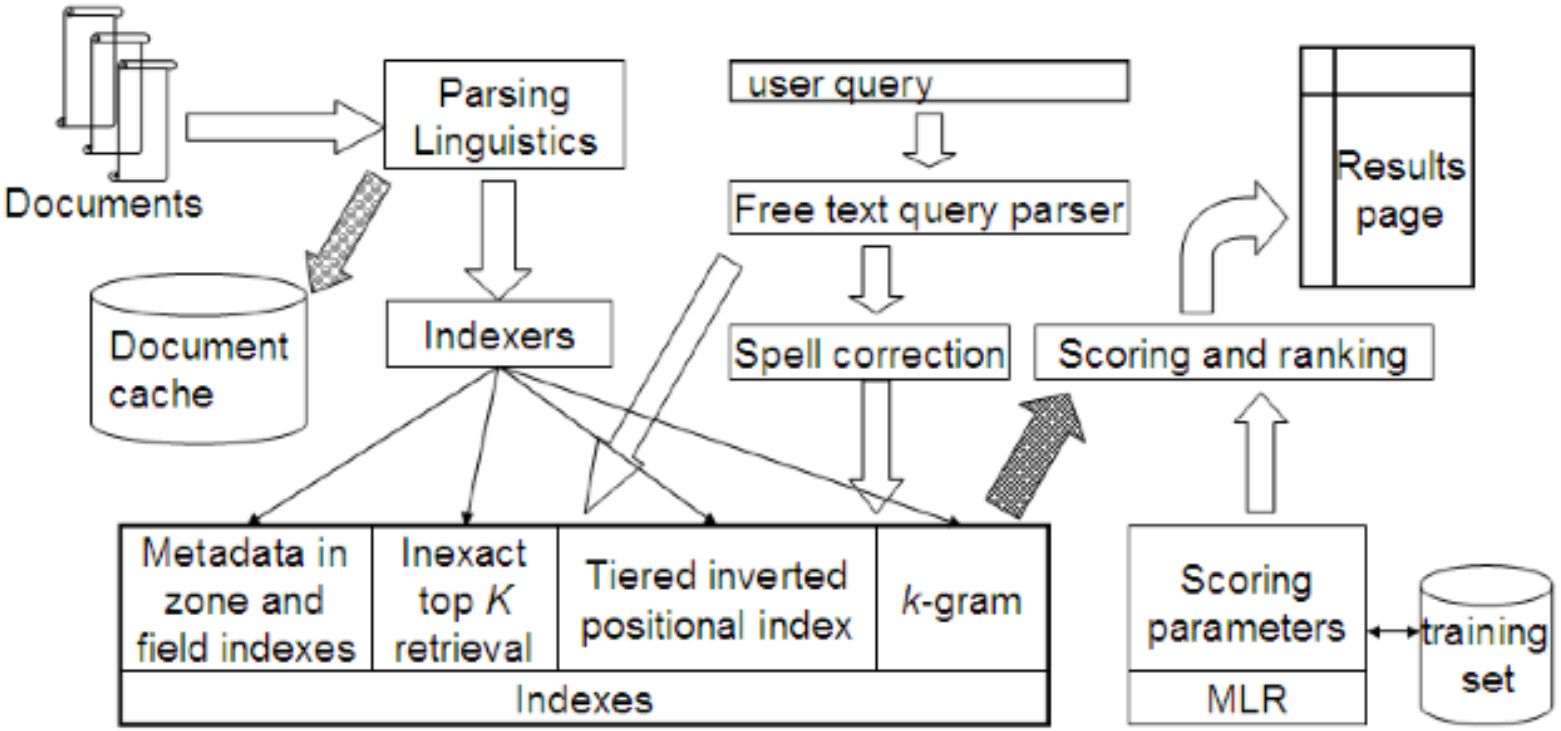
مبانی بازیابی اطلاعات و جستجوی وب

Scores in a Complete Search System ۷-

Original slides by: Hinrich Schütze and Christina Lioma

Complete search system



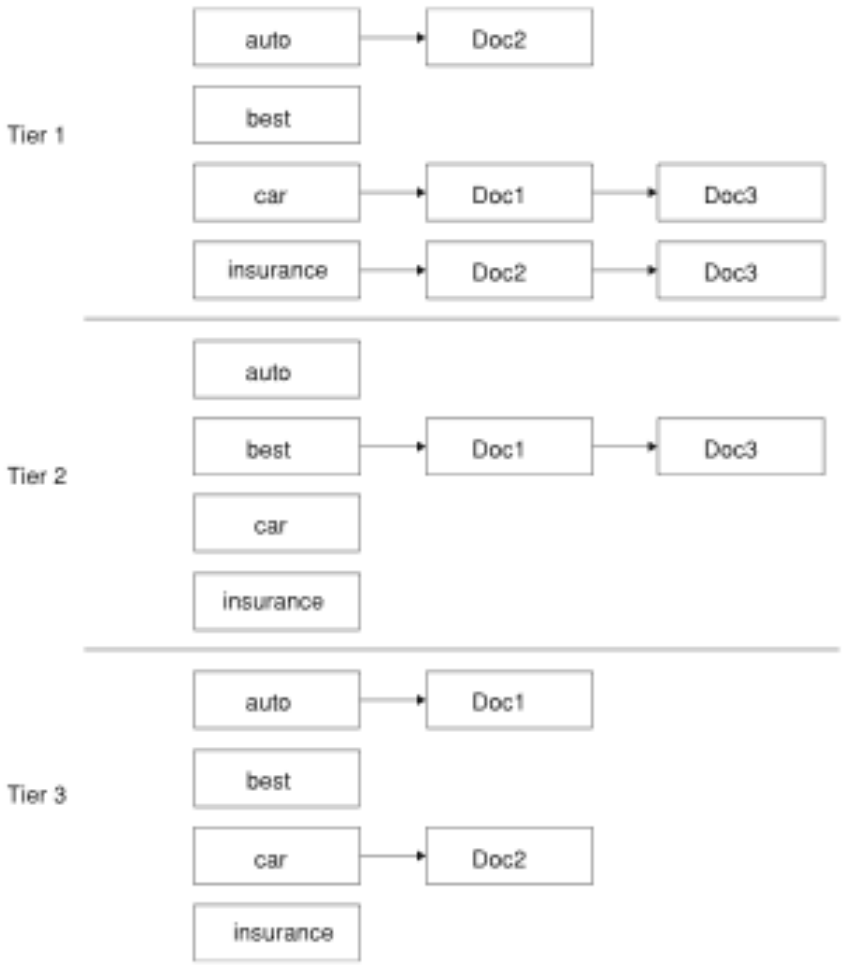
2

Tiered indexes

- Basic idea:
  - Create several tiers of indexes, corresponding to importance of indexing terms
  - During query processing, start with highest-tier index
  - If highest-tier index returns at least k (e.g., *k* = 100) results: stop and return results to user
  - If we've only found < *k* hits: repeat for next index in tier cascade
- Example: two-tier system
  - Tier 1: Index of all titles
  - Tier 2: Index of the rest of documents
  - Pages containing the search words in the title are better hits than pages containing the search words in the body of the text.

3

Tiered index



Tiered indexes

- The use of tiered indexes is believed to be one of the reasons that Google search quality was significantly higher initially (2000/01) than that of competitors.
- (along with PageRank, use of anchor text and proximity constraints)

5

Components we have introduced thus far

- Document preprocessing (linguistic and otherwise)
- Positional indexes
- Tiered indexes
- Spelling correction
- *k*-gram indexes for wildcard queries and spelling correction
- Query processing
- Document scoring
- Term-at-a-time processing

6

Components we haven't covered yet

- Document cache: we need this for generating snippets (=dynamic summaries)
- Zone indexes: They separate the indexes for different zones: the body of the document, all highlighted text in the document, anchor text, text in metadata fields etc
- Machine-learned ranking functions
- Proximity ranking (e.g., rank documents in which the query terms occur in the same local window higher than documents in which the query terms occur far from each other)
- Query parser

7

Exercise: How do we compute the top *k* in ranking?

- In many applications, we don't need a complete ranking.
- We just need the top *k* for a small *k* (e.g., *k* = 100).
- If we don't need a complete ranking, is there an efficient way of computing just the top *k*?
- Naive:
  - Compute scores for all *N* documents
  - Sort
  - Return the top *k*
- What's bad about this?
- Alternative?

8

Use min heap for selecting top *k* ouf of *N*

- Use a binary min heap
- A binary min heap is a binary tree in which each node's value is less than the values of its children.
- Takes $O(N \log k)$ operations to construct (where N is the number of documents) . . .
- . . . then read off k winners in $O(k \log k)$ steps

9

Heuristics for finding the top *k* even faster

- Document-at-a-time processing
  - تکمیل محاسبات شباهت پرس و جو-سند برای سند d_i قبل از محاسبه شباهت پرس و جو-سند برای سند d_{i+1}
  - نیازمند ترتیب سازگار اسناد در لیست های پست
- Term-at-a-time processing
  - We complete processing the postings list of query term t_i before starting to process the postings list of t_{i+1}.
  - Requires an accumulator for each document "still in the running"
- The most effective heuristics switch back and forth between term-at-a-time and document-at-a-time processing.

10

Inexact top K document retrieval

- در بسیاری از برنامه ها بازیابی تعداد k سند که امتیاز آنها خیلی نزدیک امتیاز k سند برتر است، کافی است. بنابراین میتوان از محاسبه امتیاز بسیاری از اسناد جلوگیری کرد. راهکارهای مختلفی برای این امر وجود دارد.

11

Inexact top K document retrieval

- Only consider documents containing terms whose idf exceeds a preset threshold. Thus, in the postings traversal, we only traverse the postings for terms with high idf
- We only consider documents that contain many (and as a special case, all) of the query terms.

12

Inexact top K document retrieval

- Precompute, for each term t in the dictionary, the set of the r documents with the highest weights for t. 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.
- Now, given a query q we take the union of the champion lists for each of the terms comprising q. We now restrict cosine computation to only these documents in A.

13

Non-docID ordering of postings lists

- So far: postings lists have been ordered according to docID.
- Alternative: a query-independent measure of "goodness" of a page
- Example: PageRank $g(d)$ of page d, a measure of how many "good" pages hyperlink to d (chapter 21)
- Order documents in postings lists according to PageRank: $g(d_1) > g(d_2) > g(d_3) > \ldots$
- Define composite score of a document:

$$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

- This scheme supports early termination: We do not have to process postings lists in their entirety to find top k.

14

Impact ordering

- در تمامی لیستهای پست قبلی، ترتیب اسناد مشترک بود. مثلا بر اساس شناسه سند یا بر اساس امتیازهای کیفی ایستا
- بنابراین میتوان از document-at-atime استفاده کرد.
- ترتیب متفاوت لیست های پست؟

منابع

- فصل ششم و هفتم کتاب An introduction to information retrieval