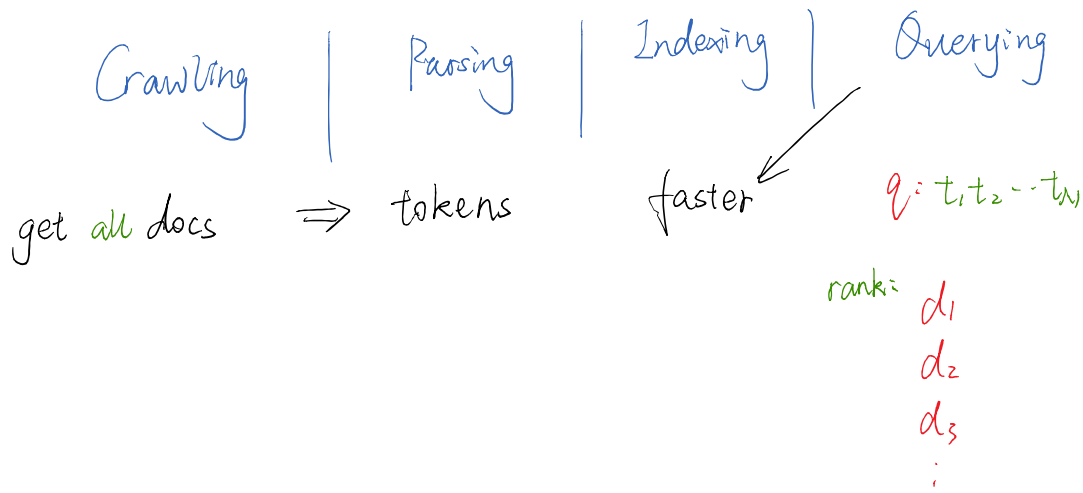
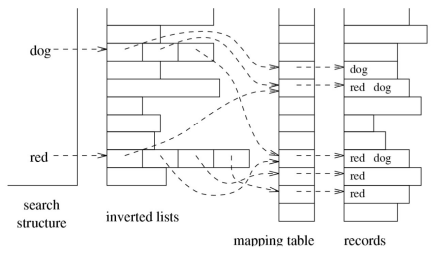
W4

2016年9月3日 19:51



Web search
- **Crawling**: the data to be searched needs to be gathered from the web.
  - challenge
    - no central index of URLs of interest
    - crawler trap
    - The Robots Exclusion Standard
- **Parsing**: the data then needs to be translated into a canonical form.
  - determining the format of the page
    - Pdf?
    - character decoding
  - discarding metadata and hidden information
  - Tokenisation
    - Fold case
    - Spilt on whitespace
    - Remove stopwords
  - canonicalisation (dates, punctuations, num, spelling, usage)
    - Stemming
    - lemmatisation
  - Remove punctuations
  - Zoning (meta data/scraping)
    - May remove or keep
      - favour pages that have the query terms in titles
- **Indexing**: data structures must be built to allow search to take place efficiently.
  - For speed
  - inverted file
    - Why call it? (d,t) -> (t,d)
    - Structure
      - For each distinct word t, the search structure contains
        - A pointer to the start of the corresponding inverted list
          - For each distinct word t, the inverted list contains:
            - The identifiers d of documents containing t, as ordinal numbers
            - The associated frequency f_{d,t} of t in d. (We could instead store w_{d,t} or w_{d,t}/W_d.)
        - A count f_t of the documents containing t.



How (same structure is used for Boolean and ranked querying)
- To evaluate a general Boolean query,
  - Fetch the inverted list for each query term.
  - Use intersection of lists to resolve AND.
    - For strictly conjunctive queries, query processing should start with the shortest list
  - Use union of lists to resolve OR.
  - Take the complement of a list to resolve NOT
  - Ignore within-document frequencies.
- To evaluate a query under the cosine measure

1. Allocate an accumulator A_d for each document d, and set $A_d \leftarrow 0$.
2. For each query term t,
   1. Calculate $w_{q,t}$, and fetch the inverted list for t.
   2. For each pair $\langle d_i, f_{d,t} \rangle$ in the inverted list
      Calculate $w_{d,t}$, and
      Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$.
3. Read the array of W_d values and, for each $A_d > 0$,
   Set $A_d \leftarrow A_d / W_d$.
4. Identify the r greatest A_d values and return the corresponding documents.

Accumulator costs
- only low f_t (rare) terms are allowed to create accumulators
  - Limit accumulators num
    - Limiting approach

1. Create an empty set A of accumulators.
2. For each query term t, ordered by decreasing $w_{q,t}$
   1. Calculate $w_{q,t}$, and fetch the inverted list for t.
   2. For each pair $\langle d_i, f_{d,t} \rangle$ in the inverted list
      If there is no accumulator for d and $|A| < L$,
      create an accumulator A_d for d.
      If d has an accumulator calculate $w_{d,t}$ and
      set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$.
   3. For each accumulator set $A_d \leftarrow A_d / W_d$.
   4. Identify the r greatest A_d values and return these documents.

    - Thresholding approach

1. Create an empty set A of accumulators, and set a threshold S.
2. For each query term t, ordered by decreasing $w_{q,t}$
   1. Calculate $w_{q,t}$, and fetch the inverted list for t.
   2. For each pair $\langle d_i, f_{d,t} \rangle$ in the inverted list
      Calculate $w_{d,t}$.
      If there is no accumulator for d and $w_{q,t} \times w_{d,t} > S$,
      create an accumulator A_d for d.
      If d has an accumulator
      set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$.
   3. For each accumulator set $A_d \leftarrow A_d / W_d$.
   4. Identify the r greatest A_d values and return these documents.

- **Querying**: the data structures must be processed in response to queries.
  - Phrase queries
    - three main strategies
      - Process queries as bag-of-words, so that the terms can occur anywhere in matching documents, then post-process to eliminate false matches.
      - Add word positions (word counts, not byte counts) to the index entries, so the location of each word in each document can be used during query evaluation.
        $\langle d, f_{d,t} \rangle \longrightarrow \langle d, f_{d,t}, p_1, \ldots, p_{f_{d,t}} \rangle$
      - Use some form of phrase index or word-pair index so that they can be directly identified without using the inverted index
    - Evaluation
      - Similarity can be computed (phrase can be treated as an ordinary term)
        - But first necessary to use the inverted lists for the terms in the phrase to construct an inverted list for the phrase itself
          - requires the index to be extended to include word positions & in-document frequency
      - Neglect common words and make use of word position
      - Or Proximity search
        - build a complete index of two-word phrases
          - Favour documents where the terms are near to each other.
          - Search for “phrases” where the terms are within a specified distance of each other.

50% web

What's more
- Weight. Related to ranking.
  - Link analysis
    - HITS (hyperlinked-induced topic search)
      - Beyond subj
    - PageRank
      - in most cases the importance of PageRank is low.
      - Anchor text, however, is crucial