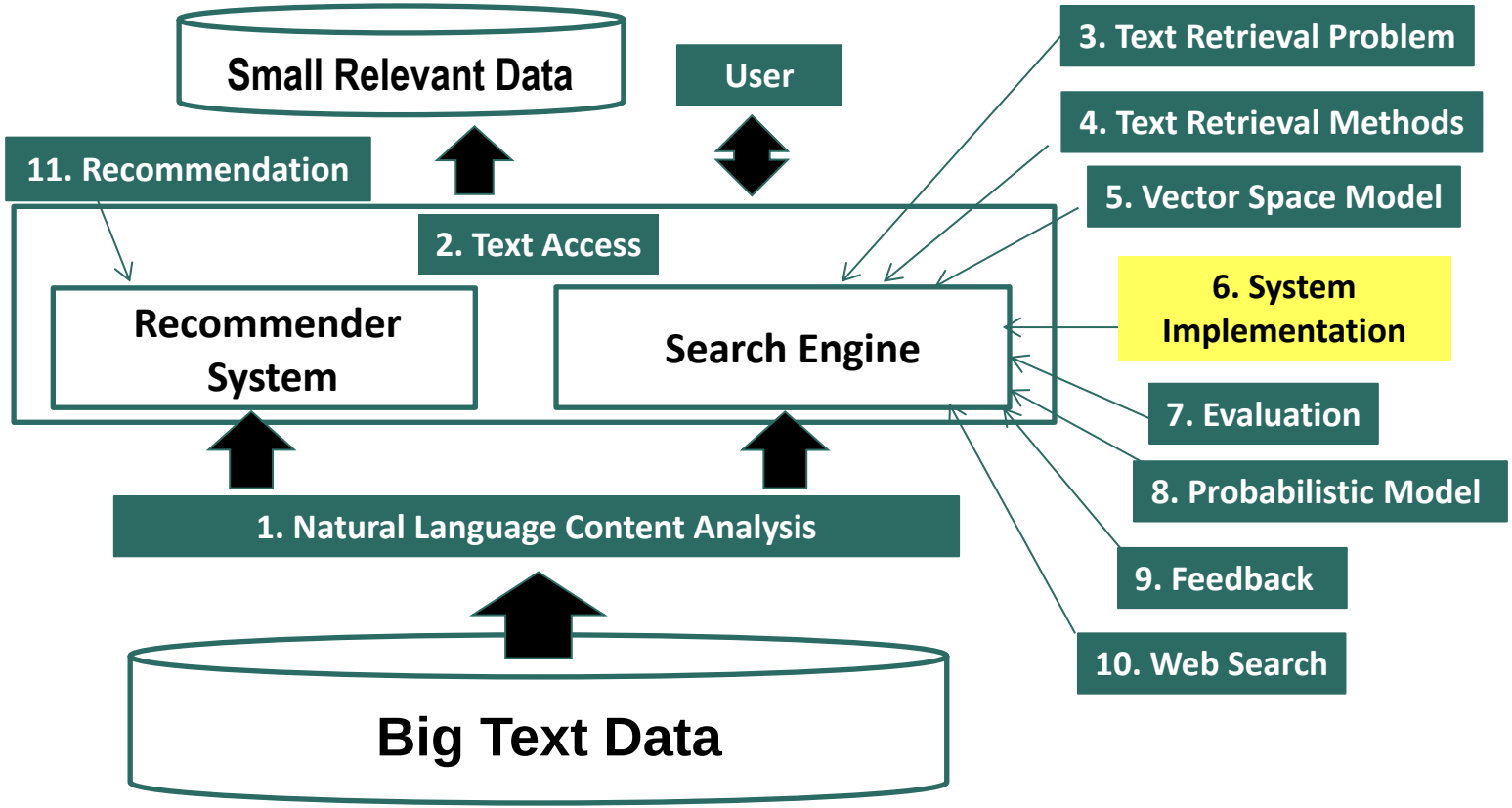
Text Retrieval and Search Engines

System Implementation: Inverted Index Construction

ChengXiang "Cheng" Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

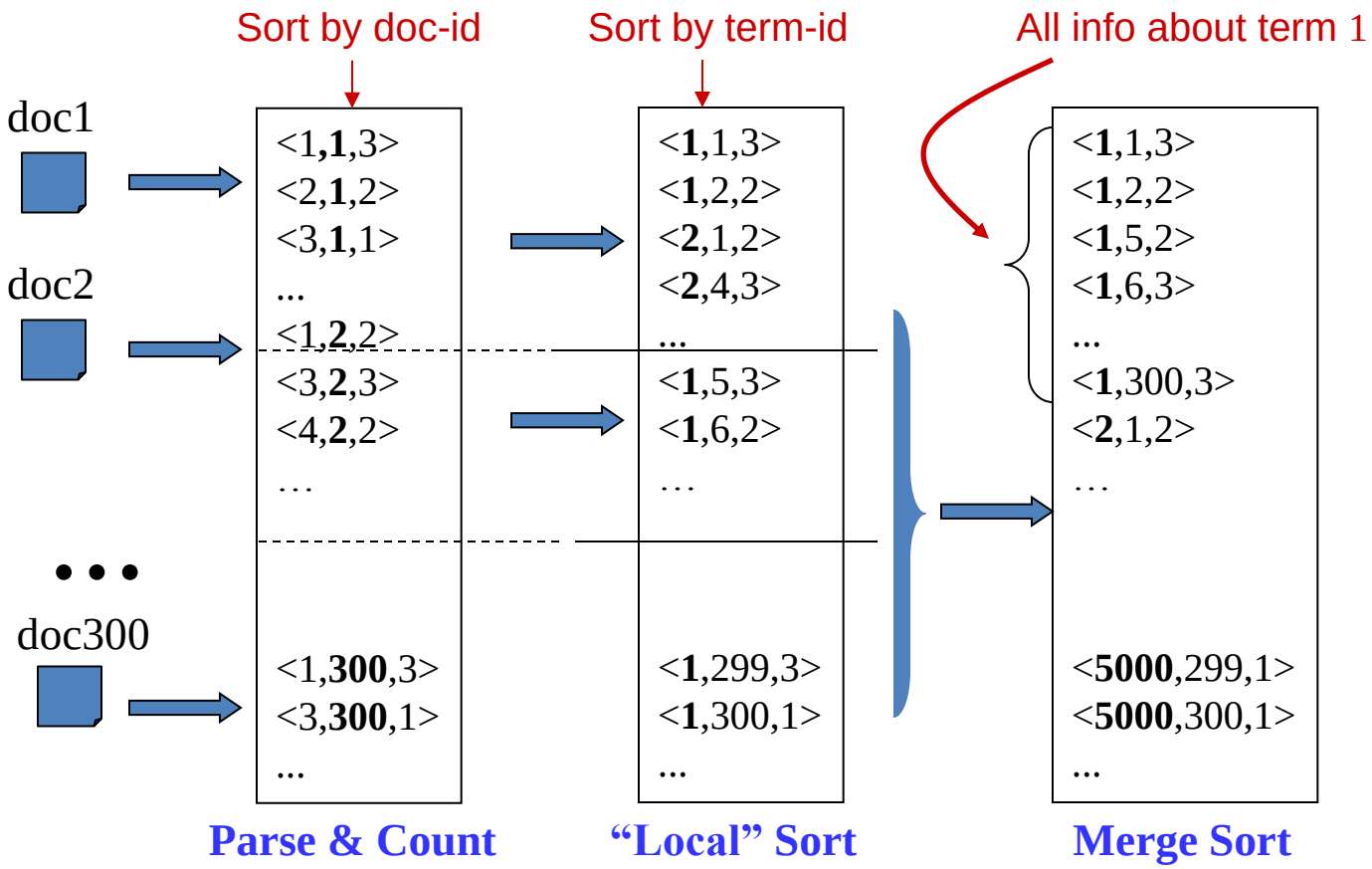
System Implementation: Inverted Index Construction



Constructing Inverted Index

- The main difficulty is to build a huge index with limited memory
- Memory-based methods: not usable for large collections
- Sort-based methods:
  - Step 1: Collect local (termID, docID, freq) tuples
  - Step 2: Sort local tuples (to make “runs”)
  - Step 3: Pair-wise merge runs
  - Step 4: Output inverted file

Sort-based Inversion



Term Lexicon:

the 1
campaign 2
news 3
a 4
…

DocID Lexicon:

doc1 1
doc2 2
doc3 3
…

Inverted Index Compression

- In general, leverage skewed distribution of values and use variable-length encoding
- TF compression
  - Small numbers tend to occur far more frequently than large numbers (why?)
  - Fewer bits for small (high frequency) integers at the cost of more bits for large integers
- Doc ID compression
  - “d-gap” (store difference): d1, d2-d1, d3-d2,…
  - Feasible due to sequential access
- Methods: Binary code, unary code, γ-code, δ-code, …

Integer Compression Methods

- Binary: equal-length coding
- Unary: $x \geq 1$ is coded as x-1 one bits followed by 0, e.g., 3=> 110; 5=>11110
- γ-code: x=> unary code for $1+\lfloor \log x \rfloor$ followed by uniform code for $x-2^{\lfloor \log x \rfloor}$ in $\lfloor \log x \rfloor$ bits, e.g., 3=>101, 5=>11001
- δ-code: same as γ-code ,but replace the unary prefix with γ-code. E.g., 3=>1001, 5=>10101

Uncompress Inverted Index

- Decoding of encoded integers
  - Unary decoding: count 1's until seeing a zero
  - γ-decoding
    - first decode the unary part; let value be k+1
    - read k more bits decode them as binary code; let value be r
    - the value of the encoded number is $\mathbf{2^k+r}$
- Decode doc IDs encoded using d-gap
  - Let the encoded ID list be x1, x2, x3, ….
  - Decode x1 to obtain doc ID1; then decode x2 and add the recovered value to the doc ID1 just obtained
  - Repeatedly decode x3, x4, …., and the recovered value to the previous doc ID.