COMP90042 Web Search & Text Analysis

Workshop Week 4

Zenan Zhai

March 26, 2019

University of Melbourne

Road Map

Indexing

- Data Structure
  - Document-Term Matrix
  - Inverted Index
- Compression
  - Variable Byte Compression
  - OptPFor Delta Compression
- Index Construction
  - Invert Batch Indexing
  - Auxiliary Indexing
  - Logarithmic Indexing

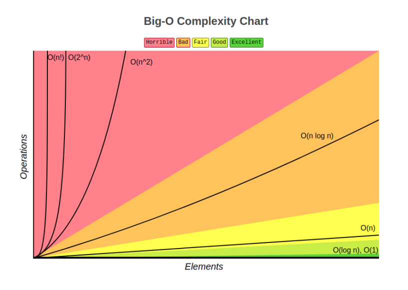
Road Map

Search

- Vector Space Models
    - TF-IDF
    - BM25
- Efficient Query Processing
    - Operation GEQ
    - WAND
- Query Completion
    - Prefix Trie
    - Range Maximum Query
- Query Expansion
    - Relevance Feedback
    - Semantic-Based Methods
- Phrase Search
    - Inverted Index + Positional Information
    - Suffix Array
- Evaluation and Re-rank

Warm Up - Complexity

Time Complexity



Big O cheat sheet

Notation

- $T(n) = O(f(n)) \Leftrightarrow \exists\ c, n_o, \forall\ n > n_o, T(n) \leq c \cdot O(f(n))$
- $T(n) = \Omega(f(n)) \Leftrightarrow \exists\ c, n_o, \forall\ n > n_o, T(n) \geq c \cdot \Omega(f(n))$
- $T(n) = \Theta(f(n)) \Leftrightarrow \exists\ c_1, c_2, n_o, \forall\ n > n_o,$
  $c_1 \cdot \Theta(f(n)) \leq T(n) \leq c_2 \cdot \Theta(f(n))$

Warm Up - Complexity

Space Complexity

- Amount of auxiliary space the algorithm need in the function of input size.

Example - Merge Sort

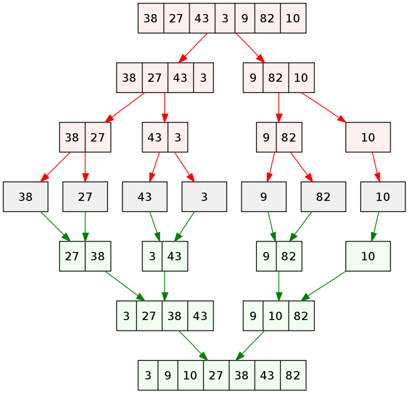
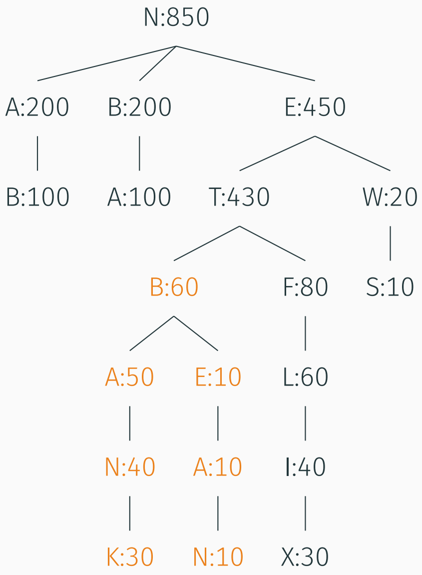


Fig from Wikipedia.

Outline

- Query Completion
    - Prefix Trie
    - Range Maximum Query
- Query Expansion
    - Relevance Feedback
    - Semantic-Based Methods
- Phrase Search
    - Inverted Index + Positional Information
    - Suffix Array
- Evaluation
    - Mean Average Precision (MAP)
    - Rank-biased Precision (RBP)
- Re-rank
    - Point-wise learning
    - Pair-wise learning

Prefix Trie - Data Structure



- Each edge a character in a prefix.
- Each node store the frequency of the prefix being searched.
- Children of nodes are ordered.
- Traverse the tree to generate an array.
- Sub-tries are continuous sub-arrays.

60	50	40	30	10	10	10

Array for sub-trie in red.

Range Maximum Query (RMQ)

How to get top-k most frequent items have the given prefix?

4	9	10	5	32	7	13	21	1

- Sort the corresponding sub-array and take the first-k elements.
- Less time complexity?
- Less space complexity?

RMQ - Reduced Time Complexity

Pre-compute max-value for all sub-arrays with $O(N^2)$ space.$M = [m_{ij}]_{N\times N}$
$RMQ(i,j)$ can be done by accessing the value m_{ij} in matrix.

$arr \leftarrow Array[0...k-1]$;
$heap \leftarrow emptyMaxHeap()$;
$heap.insert(left, right)$;
for *i in [0...k-1]* **do**
    $node \leftarrow heap.pop()$;
    $maxPos \leftarrow RMQ(node.left, node.right)$;
    $arr[i] \leftarrow maxPos$;
    if $node.left \neq null$ **then**
        $heap.insert(node.left, maxPos - 1)$;
    end
    if $node.right \neq null$ **then**
        $heap.insert(maxPos + 1, node.right)$;
    end
end

RMQ - Reduced Space Complexity

- Store max-value of every $A[i, i + 2^n]$ only, instead of all $A[i, j]$.
- Use 2 overlapping region $A[i, P]$ and $A[Q, j]$ to cover the region in query $A[i, j]$.
- Take $max(RMQ(i, P), RMQ(Q, j))$.

Outline

- Query Completion
  - Prefix Trie
  - Range Maximum Query
- Query Expansion
  - Relevance Feedback
  - Semantic-Based Methods
- Phrase Search
  - Inverted Index + Positional Information
  - Suffix Array
- Evaluation
  - Mean Average Precision (MAP)
  - Rank-biased Precision (RBP)
- Re-rank
  - Point-wise learning
  - Pair-wise learning

Query Expansion

Feedback-based

- User Relevance Feedback
- Pseudo Relevance Feedback
- Indirect Relevance Feedback

Questions:

- Can we expand query without feedback?

Outline

- Query Completion
  - Prefix Trie
  - Range Maximum Query
- Query Expansion
  - Relevance Feedback
  - Semantic-Based Methods
- Phrase Search
  - Inverted Index + Positional Information
  - Suffix Array
- Evaluation
  - Mean Average Precision (MAP)
  - Rank-biased Precision (RBP)
- Re-rank
  - Point-wise learning
  - Pair-wise learning

Positional Inverted Index

	DocID	*Frequency*	*Position*
big	< 1, 3, 5>	< 1, 2, 1>	< <23>, <43, 65>, <31> >
brother	< 2, 3, 6>	< 1, 1, 1>	< <2>, <42>, <67> >
...			

Query: big brother

- Intersect *DocID* first, then intersect *Position*.
- Sort list by length, starting from the smallest.

	DocID	*Frequency*	*Position*
big brother	<3>	<1>	< <42, 43> >
...			

String Matching and Suffix Array

Trivial string matching takes $O(|n| \cdot |m|)$ for matching m in n.

Suffix arrays of T (abrac$).

id	start	suffix array
0	5	$abrac
1	0	abrac$
2	3	ac$abr
3	1	brac$a
4	4	c$abra
5	2	rac$ab

- Store suffix array
  - Complete arrays $O(n^2 log\sigma)$.
  - Start index only $O(nlogn)$
- Sort n array of length n takes
  - $O(n^2 logn)$ for quick sort
  - $O(n)$ *(Li et. al., 2016)*
- Perform binary search
  - $T(n) = O(m \cdot logn)$
  - $S(n) = O(|T| + nlogn)$

Outline

- Query Completion
  - Prefix Trie
  - Range Maximum Query
- Query Expansion
  - Relevance Feedback
  - Semantic-Based Methods
- Phrase Search
  - Inverted Index + Positional Information
  - Suffix Array
- Evaluation
  - Mean Average Precision (MAP)
  - Rank-biased Precision (RBP)
- Re-rank
  - Point-wise learning
  - Pair-wise learning

Evaluation Metric

Recap:

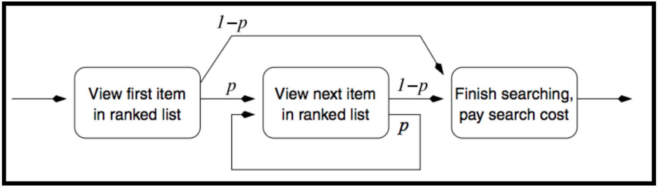
- Precision: $\frac{TP}{TP+FP}$
- Recall: $\frac{TP}{TP+FN}$
- F_1 measure: $\frac{2 \cdot P \cdot R}{P+R}$

IR measurements:

- Based on relevance vector
- Precision@k: only first k element in relevance vector.
- AP: average P@k for all k, $r_k = 1$.
- MAP: average AP for all queries.

Rank-biased Precision

Introducing patience factor p to average precision.



$$RBP = \sum_{i=1}^{d} r_i \times p^{i-1} \times (1-p)$$

- r_i: the i^{th} element in relevance vector.
- p^{i-1}: probability of the reader reaches the i^{th} element.
- $(p-1)$: probability of the reader stops at this element.
- Assumption: $P(Stop)$ independent to i.

Outline

- Query Completion
  - Prefix Trie
  - Range Maximum Query
- Query Expansion
  - Relevance Feedback
  - Semantic-Based Methods
- Phrase Search
  - Inverted Index + Positional Information
  - Suffix Array
- Evaluation
  - Mean Average Precision (MAP)
  - Rank-biased Precision (RBP)
- Re-rank
  - Point-wise learning
  - Pair-wise learning

Learning to Rank

Point-wise

- Predict relevance factor $P(r_i|x_i)$ (e.g. [-2, 2]).
- Sort documents by relevance factor.

Pair-wise

- Predict which document is more relevant $P(y_{i,j}|x_i, x_j)$
- Sort by comparing documents.

How to convert documents to X?