Information Retrieval
Topic-Tolerant Retrieval
(Hashes, Trees)
Lecture-9

Prepared By

Dr. Rasmita Dash & Dr. Rasmita Rautray
Associate Professor
Dept. of CSE

Content

- Dictionaries
- Data structure for dictionaries
    - Hash
    - Tree

Dictionaries

- Dictionary: the data structure for storing the
- term vocabulary Term vocabulary: the data
- For each term, we need to store a couple of items:
    - ➢ document frequency
    - ➢pointer to postings list
- How do we look up a query term q_i in the dictionary at query time?

Data structures for looking up terms

- Two different types of implementations: hashes and search trees.
- Some IR systems use hashes, some use search trees.
- Criteria for when to use hashes vs. search trees:
    - How many terms are we likely to have?
    - Is the number likely to remain fixed, or will it keep growing?
    - What are the relative frequencies with which various terms will be accessed?

Hashes

- Hash table: an array with a hash function
    - ➢Input key; output integer: index in array.
    - ➢ Hash function: determine where to store / search key.
    - ➢Hash function that minimizes chance of collisions
        - Use all info provided by key (among others).
- Each vocabulary term (key) is hashed into an integer.
- At query time: hash each query term, locate entry in array

Pros and Cons of Hashes

Pros:

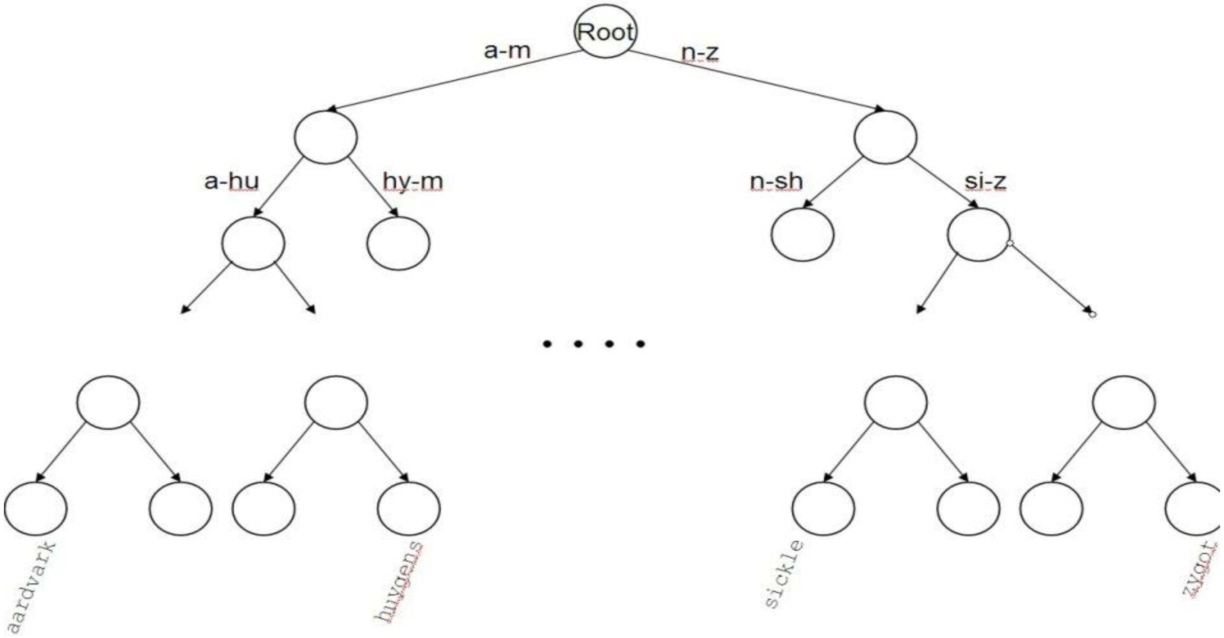
- Lookup in a hash is faster than lookup in a tree. (Lookup time is constant.)

Cons:

- No easy way to find minor variants

  (resume vs. r´esum´e)
- No prefix search (all terms starting with automat)
- Need to rehash everything periodically if vocabulary keeps growing
- Hash function designed for current needs may not suffice in a few years’ time

Search trees



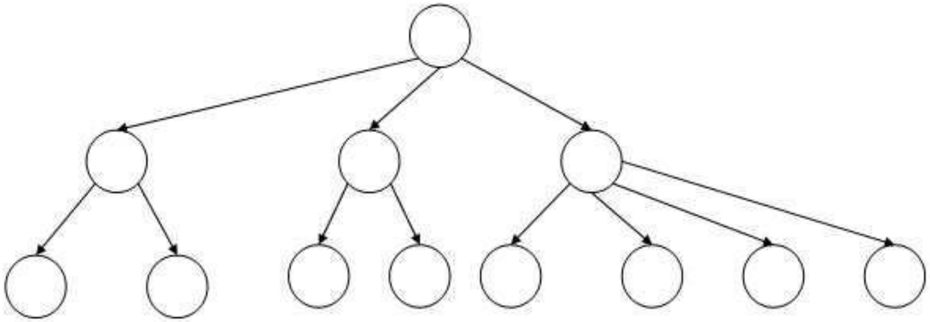
- Partition vocabulary terms into two subtrees, those whose first letter is between a and m, and the rest (actual terms stored in the leafs).
- Anything that is on the left subtree is smaller than what’s on the right.
- Trees solve the prefix problem (find all terms starting with automat).

Binary search tree

- Cost of operations depends on height of tree.
- Keep height minimum / keep binary tree balanced: for each node, heights of subtrees differ by no more than 1.
- O(log M) search for balanced trees, where M is the size of the vocabulary.
- Search is slightly slower than in hashes But: re-balancing binary trees is expensive (insertion and deletion of terms).

B-tree

- Need to mitigate re-balancing problem – allow the number of sub-trees under an internal node to vary in a fixed interval.
- B-tree definition: every internal node has a number of children in the interval [a, b] where a, b are appropriate positive integers, e.g., [2, 4].



Thank You