IN[34]120 Repetition lecture

2023-09-20 10:15 @ Prolog

Partie de: Oliver
(From group 1, fridays 10:15 @ Chill)
((These slides are stolen from the earlier seminars))

Thèmes
- Terms
- Posting lists:
- Inverted indeces
- Suffix arrays
- Permuterm indeces
- Tries
- Aho-Corasick

Terms

- (Someone ™ wanted this covered)
- Terms may be thought of as a "word"
- *It's not that deep*

Stop words

- Processing terms is expesnive!
- We want to process fewer terms
- Possible: Remove non-nessescary terms
- Semantic value
- "the", "a", "an"

Posting list

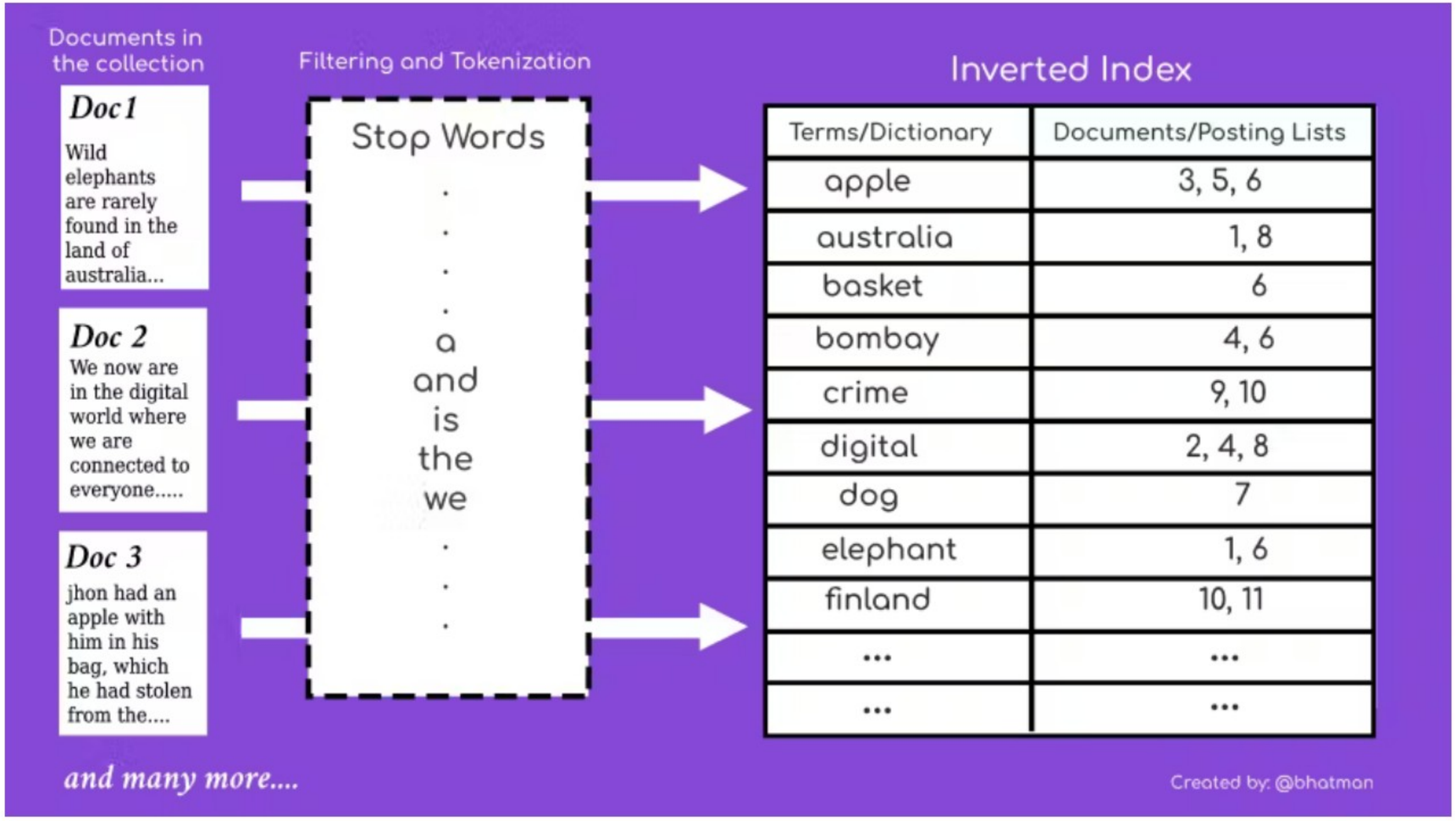
- A set of documents
- All the docs in a PL contain the same term
- "here are alle the docs containing the term 'Heracles'"

Posting lists cont.

- Optimization: Numbers (DocIDs)
- 1 - 4 - 6 - 9
- NB: List must be sorted

Inverted index

- Mapping: term -> posting list
- Like the register of a book



Visualization: inverted index with posting lists. No stop words.

We now have: Primitive search

- → No ranking (Boolean relevance)
- → No tolerance (terms must be spelled 100% corretly)

Why must the posting lists be sorted?

The correct answer is: So that we may employ efficient algorithms on them

Operations on posting lists

- → Union (shared postings)
- → Intersection (exclusive postings)
- → For now: only 2 lists
- → Later (Assignement C): merge n lists

Suffix arrays

- → Data structure for search
- → Find matching terms from suffixes
- → Sorted lexigraphically - why?

Suffix Array Example

Given String: banana

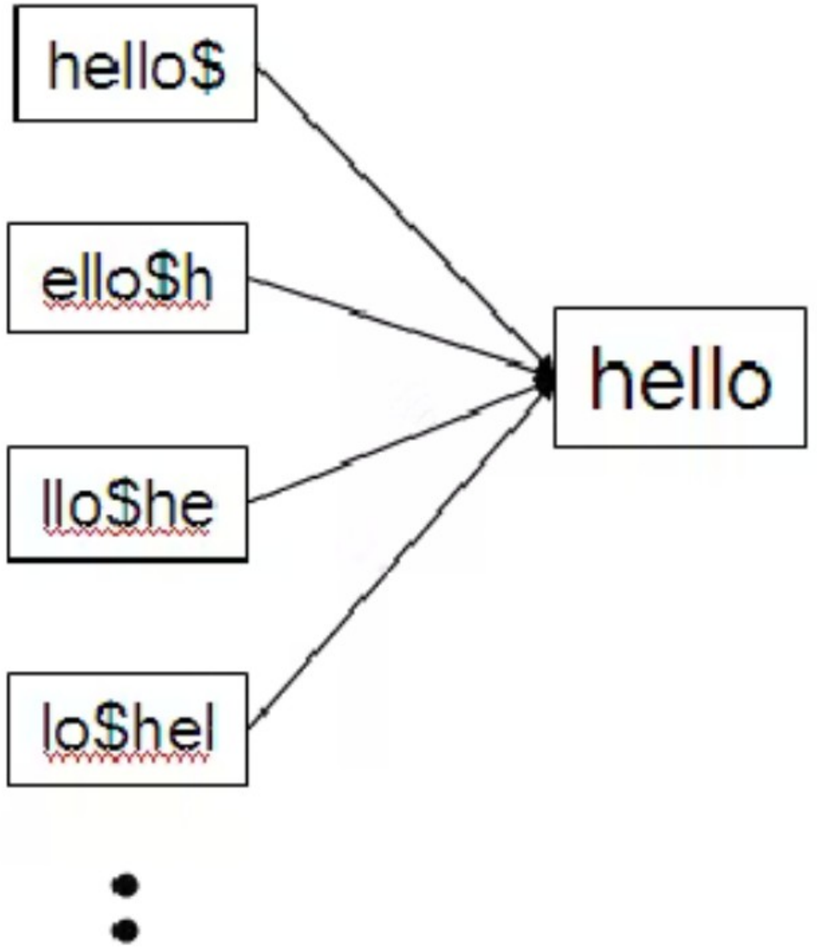
Suffixes		Sorted Suffixes
0 banana		5 a
1 anana	Sort the Suffixes	3 ana
2 nana	---------->	1 anana
3 ana	alphabetically	0 banana
4 na		4 na
5 a		2 nana

Suffix array: {5, 3, 1, 0, 4, 2}

Visualization of a basic suffix array for 1 term

Permuterm indeces

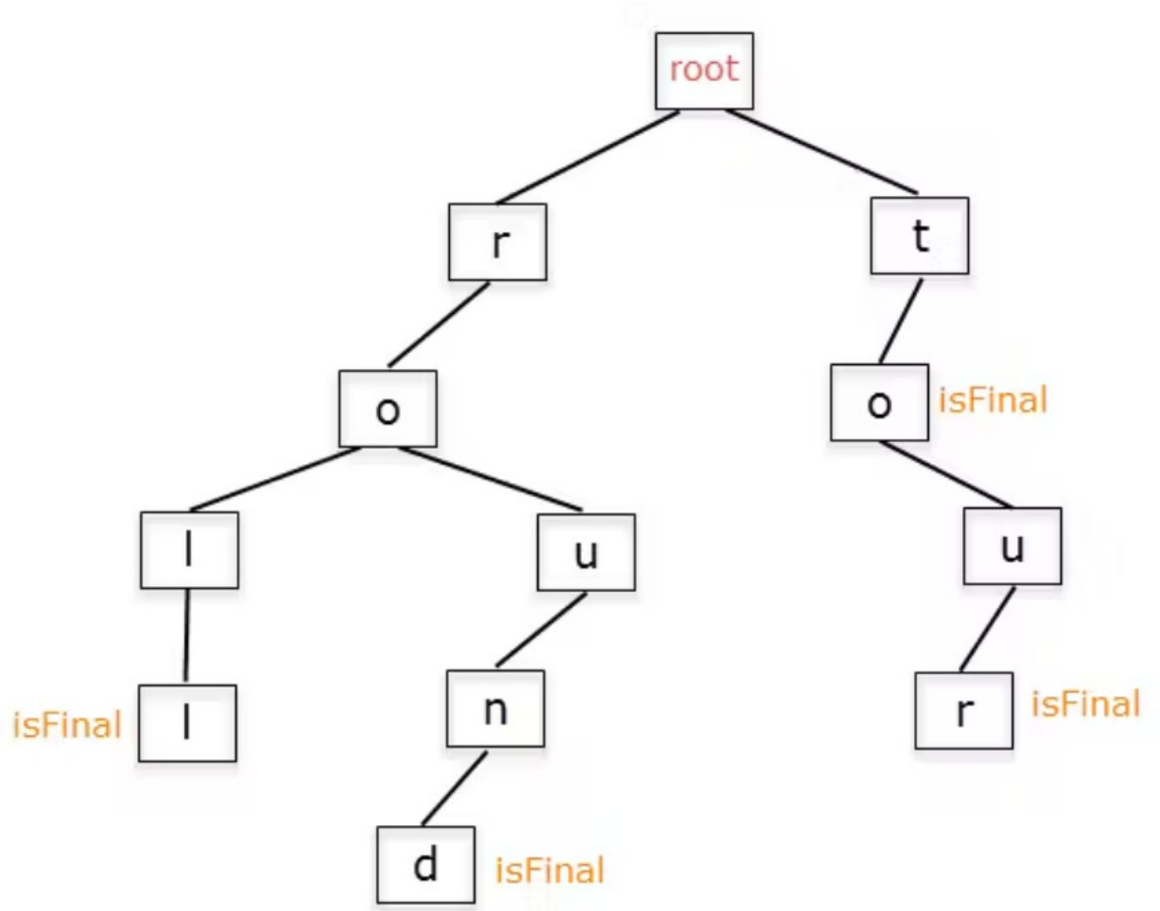
- → "permutations of the types"-index
- → Allows wildcard queries, e.g. /.ake/
- → lsh same use case as suffix arrays
- → Stores the types of the corpus "rotated"



Permuterm-index-structure for the term "hello"

Tries, Aho-Corasick algorithm

- Data structure - prefix tree. Algorithm.
- Use case: Determine if a string x appears in our corpus
- Time complexity: O(length of the string x)
- Scales irrelevant of corpus size (i.e. it's fast)



Trie. Aho-Corasick* algorithm