Monsoon 2022

Positional Indexing

- Creating Inverted Index with Term Positions

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

24th August 2021 (rajendra.2power3.com)

> Topics to be covered

- Recap:
  - Inverted Index Construction
  - Term - Document Matrix
- Boolean Information Retrieval
  - Boolean Operators
  - Boolean Retrieval
  - Boolean Queries
- Evaluation
  - Text Collection / Corpora
  - Evaluation Strategy
    - More topics to come up … Stay tuned …!!

Overview

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of web search, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Recap: Boolean Incidence Matrix

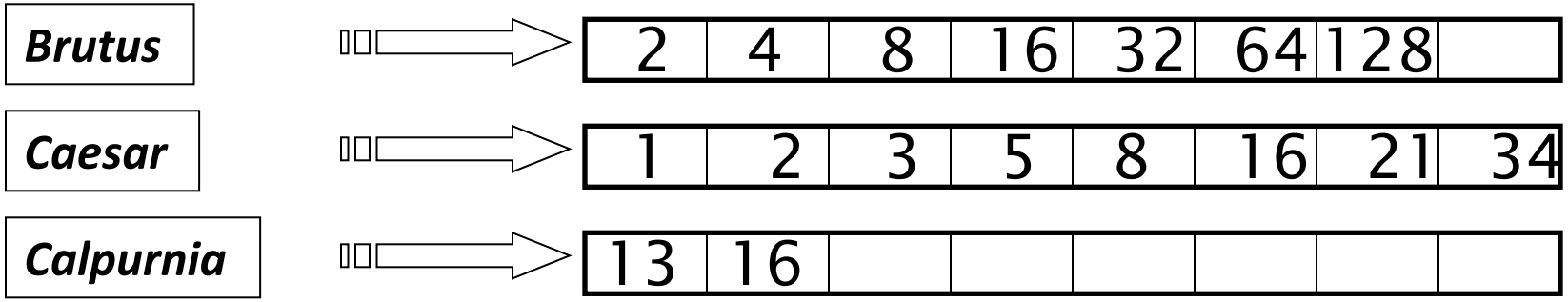
Terms	d_1	d_2	d_3	. . .	d_n
the	1	1	1	. . .	0
a	1	1	1	. . .	1
Darjeeling	1	1	1	. . .	0
is	1	1	1	. . .	0
of	1	1	1	. . .	0
in	1	0	0	. . .	1
and	1	1	0	. . .	0
Bengal	1	0	1	. . .	0
It	1	0	1	. . .	0
Its	0	1	0	. . .	1
state	1	0	1	. . .	0
West	1	0	1	. . .	1

Boolean Queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using **AND**, **OR** and **NOT** to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades
- Many search systems you still use are Boolean:
  - Email, library catalog, Mac OS X Spotlight

Query Optimization

- What is the best order for query processing?
- Consider a query that is an **AND** of n terms.
- For each of the n terms, get its postings, then **AND** them together.



Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query Processing - Exercises

- **Exercise:** If the query is ***friends*** *AND* ***romans*** *AND (NOT* ***countrymen****)*, how could we use the frequency of ***countrymen***?

- **Exercise:** Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?

- **Hint:** Begin with the case of a Boolean *formula* query: in this, each query term appears only once in the query

Phrase queries

- We want to be able to answer queries such as **"stanford university"** – as a phrase
- Thus the sentence "**I went to university at Stanford**" is not a match.
  - The concept of phrase queries has proven easily understood by users; one of the few "advanced search" ideas that works
  - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only
  - *<term : docs>* entries

A First attempt: Biword indexes

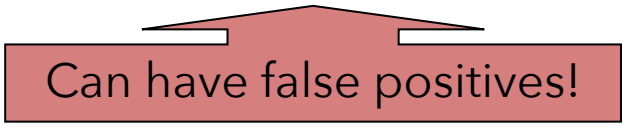
- Index **every consecutive pair of terms** in the text as a phrase
- For example the text "Friends, Romans, Countrymen" would generate the biwords
  - **friends romans**
  - **romans countrymen**
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate

Longer phrase queries

- Longer phrases can be processed by breaking them down
- ***stanford university palo alto*** can be broken into the Boolean query on biwords:

stanford university AND university palo AND palo alto

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase



Issues for Biword Indexes

- False positives, as noted before
- Index blowup due to bigger dictionary
  - Infeasible for more than Biwords, big even for them
- Biword indexes are not the standard solution (for all Biwords) but can be part of a compound strategy

Solution 2: Positional indexes

- Positional indexes are a more efficient alternative to biword indexes
- Postings lists in a nonpositional index:
  - each posting is just a docID
- Postings lists in a positional index:
  - each posting is a docID and a list of positions

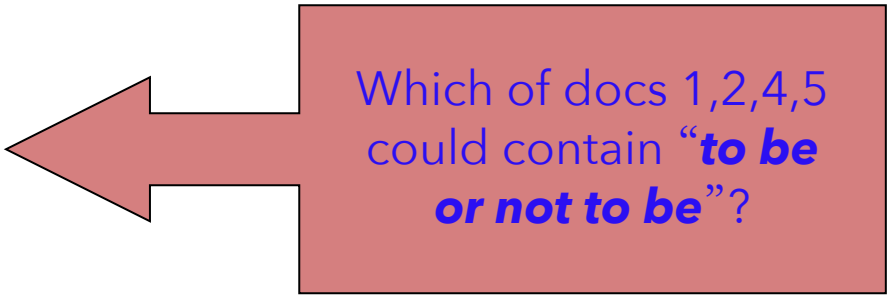
Solution 2: Positional indexes

- In the postings, store, for each ***term*** the position(s) in which tokens of it appear:

  <***term***, number of docs containing ***term***;
  doc1: position1, position2 … ;
  doc2: position1, position2 … ;
  etc.>

Positional index example

<*be*: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>



- For phrase queries, we use a merge algorithm recursively at the document level
- But we now need to deal with more than just equality

Processing a phrase query

- Extract inverted index entries for each distinct term: ***to, be, or, not***
- Merge their *doc:position* lists to enumerate all positions with "***to be or not to be***"
  - ***to***:
    - *2*:1,17,74,222,551; *4*:8,16,190,429,433; *7*:13,23,191; ...
  - ***be***:
    - *1*:17,19; *4*:17,191,291,430,434; *5*:14,19,101; ...
- Same general method for proximity searches

Positional Indexes: Example

Query: "$to_1\ be_2\ or_3\ not_4\ to_5\ be_6$"

to, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

be, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Document 4 is a match!

Positional index size

- A positional index expands postings storage substantially
- Even though indices can be compressed
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries
- used explicitly or implicitly in a ranking retrieval system

Positional index size

- Need an entry for each occurrence, not just once per document
- Index size depends on average document size
  - Average web page has < 1000 terms
  - Books, even some epic poems … easily 100,000 terms
- Consider a term with frequency 0.1%

Why?

Document size	Postings	Positional postings
1000	1	1
100,000	1	100

Rules of thumb

- A positional index is 2–4 as large as a non-positional index
- Positional index size 35–50% of volume of original text
  - Caveat: all of this holds for "English-like" languages

Proximity queries

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
  - Again, here, /*k* means "**within *k* words of**"
- Clearly, positional indexes can be used for such queries; biword indexes cannot.
- **Exercise:** Adapt the linear merge of postings to handle proximity queries
- Can you make it work for any value of *k*?
  - This is a little tricky to do correctly and efficiently

Proximity Search

- We can also use it for proximity search.
  For example: employment /4 place

- Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other

- Employment agencies that place healthcare workers are seeing growth is a hit

- Employment agencies that have learned to adapt now place healthcare workers is not a hit

Proximity Search

- Use Positional Index
- Simplest Algorithm: look at cross-product of positions of
- EMPLOYMENT in document and
- PLACE in document
- Very inefficient for frequent words, especially stop words
- Note that we want to return the actual matching positions, not just a list of documents
- This is important for dynamic summaries, etc

Combination schemes

- These two approaches can be profitably combined
  - For particular phrases ("**Michael Jackson**", "**Britney Spears**") it is inefficient to keep on merging positional postings lists
    - Even more so for phrases like "**The Who**"

- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme
  - A typical web query mixture was executed in ¼ of the time of using just a positional index
  - It required 26% more space than having a positional index alone

Combination Scheme (contd.)

- Biword indexes and positional indexes can be profitably combined
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc
- For these biwords, increased speed compared to positional postings intersection is substantial.

- **Combination Scheme:**
  - Include frequent biwords as vocabulary terms in the index
  - Do all other phrases by positional intersection

Summary

In this class, we focused on:

(a) Understanding of the basic unit of classical information retrieval systems

(b) Positional Indexes

i. Positional Index Size

(c) Proximity Search

(d) Combination Schemes

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions
It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor