https://vvtesh.sarahah.com/

Information Retrieval

Venkatesh Vinayakarao

Chennai Mathematical Institute

What we find changes who we become.

-Peter Morville.

Acknowledgment

Some slides are borrowed from the companion website of Manning et al.'s IR book
(https://nlp.stanford.edu/IR-book/)

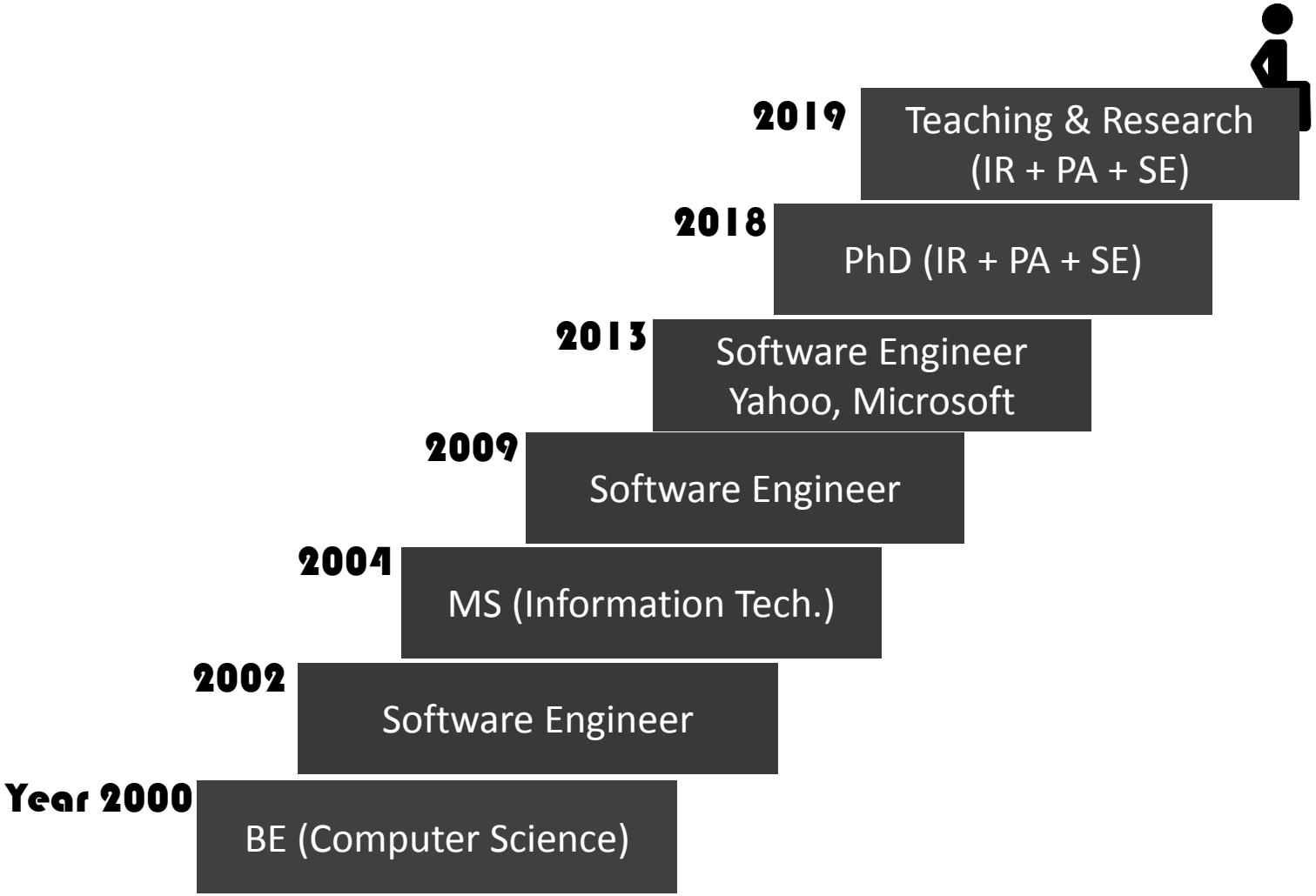
A good teacher can inspire hope, ignite the imagination, and instill a love of learning.

-Brad Henry.

Agenda

- About Me
- Introduction
- Course Dynamics
- Our First IR System
  - Linear Traversal
- Boolean Retrieval
- Evaluation

About Me


2019
Teaching & Research
(IR + PA + SE)
2018
PhD (IR + PA + SE)
2013
Software Engineer
Yahoo, Microsoft
2009
Software Engineer
2004
MS (Information Tech.)
2002
Software Engineer
Year 2000
BE (Computer Science)

Introduction

Information

Shannon's Definition, Fisher Information, Neumann Entropy, … 

Information is any entity or form that provides the answer to a question of some kind or resolves uncertainty. – Wikipedia.

Role of Information

- If only you knew
  - Which stock to invest in?
  - Which faculty to work with?
  - How to get into a top college?
  - Which course to register for?
  - What to study?
  - How to prepare for job interviews?
  - …
- If only you had the information, you could rule this world!
- What happens when all the information is deprived from you?

Solitary Confinement is Cruel

Information

Several retrieval systems: Lycos, Altavista, MSN, B idu, Yahoo!, Ask. etc.,

Google

1998

30 Trillion documents

> 130 Trillion in 2016

Digital Libraries

1970's

Universal Digital Library,

Project Gutenberg, etc.

British Library

1970's

170+ Million Collection

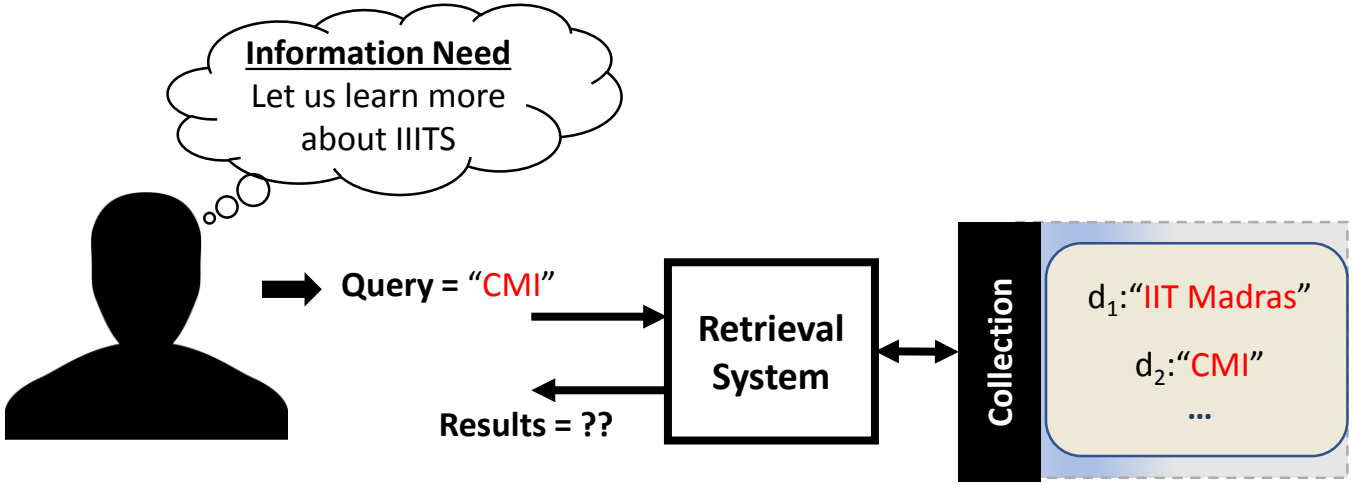
Bibliothèque nationale de France

1463

Royal Library of Alexandria

300 BC.

What is 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.

– From the Manning et al. IR Book.

Course Dynamics

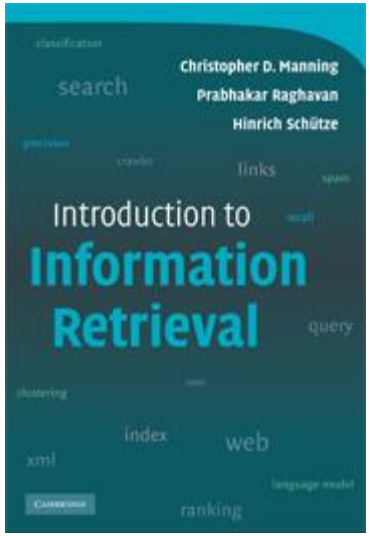
Learning Objectives

- Understand and apply text retrieval techniques to big data.
- Understand and apply text indexing techniques.
- Analyze and evaluate existing retrieval systems.

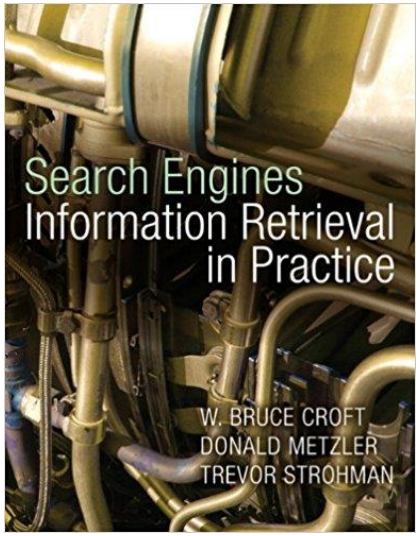
Course Website: http://vvtesh.co.in/teaching/IR-2019.html

We will use moodle for assignments.

Resources



Course Text



Reference

Evaluation

Instrument	Max Marks
Final Exam	60%
Assignments (3 * 10% each)	30%
In-Class Quiz	10%

Assignments

- May (Not necessarily though) have a programming component.
- Will test the concepts you study.
- Individual.

Exams

- Closed Book.

Office Hours

- By appointment.
  - Send me an email.
  - Find me in Room 605.
  - Keep “[IR Class]” on subject line.

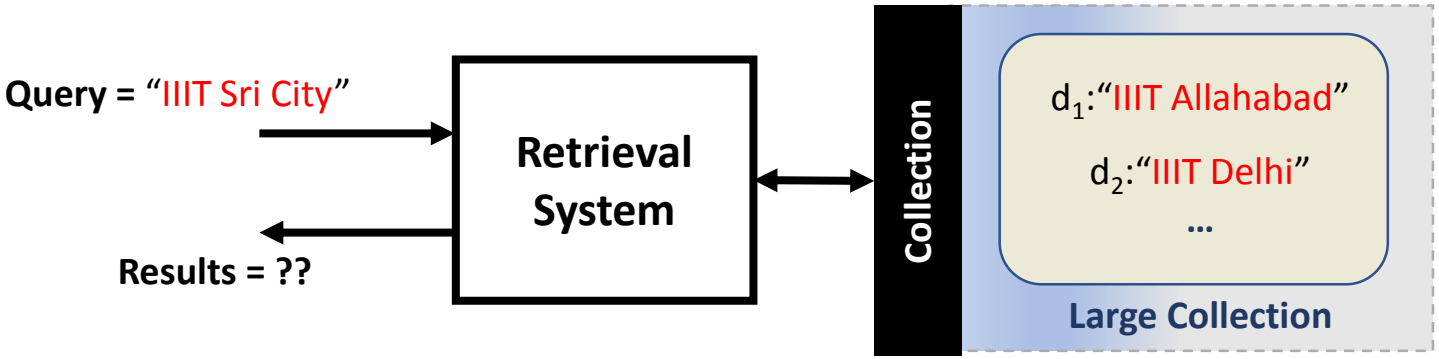
A Simple Retrieval System

Our first IR system.

Simple Retrieval Problem

- A **collection** with 5 **documents** having the following contents
  - d1: IIIT ALLAHABAD
  - d2: IIIT DELHI
  - d3: IIIT GUWAHATI
  - d4: IIIT KANCHIPURAM
  - d5: IIIT SRI CITY
- **Query** is
  - IIIT SRI CITY
- Which **document** will you match and why?

The Problem


Query = "IIIT Sri City"
Results = ??
Retrieval System
Collection
d_1:"IIIT Allahabad"
d_2:"IIIT Delhi"
...
Large Collection

One (bad) Approach

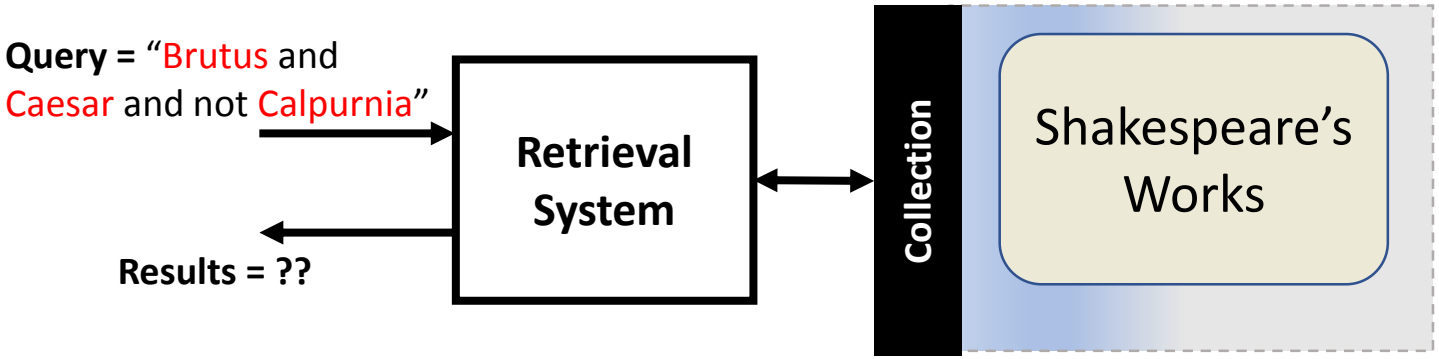
- First match the **term** IIIT.
  - Filter out documents that contain this term.
- Next match the **term** Sri.
  - Filter out documents that contain this term.
- Next match the **term** City.
  - Filter out documents that contain this term.

Three iterations!
Quiz: Can we do better?

Boolean Retrieval

Match or No-Match! No ranking of results.

Simple Conjunctive Queries



A Term-Document Incidence Matrix Example

Documents

Terms	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

"Brutus and Caesar and not Calpurnia"

Revisiting Boolean Algebra

What is the best way to get to the answer?

The Answer

"Brutus and Caesar and not Calpurnia"

1	1	0	1	0	0
1	1	0	1	1	1
1	0	1	1	1	1
AND					
1	0	0	1	0	0

Document 1 and 4 satisfy our query.

Disadvantages of term-document Matrix

- When a new document is added to collection:
  - New columns get added.
- If the collection is very large (say Millions of documents),
  - Each document has far fewer words from the dictionary.
  - So, the matrix is sparse.

Can we do better?

Instead of handling both 1s and 0s, can we only have the 1s?

Revisiting Data Structures

Arrays Vs. Linked Lists

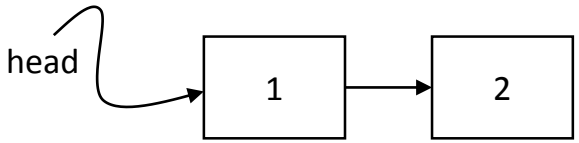
The Problem

- An n-Dimensional Vector can be represented as
  - an array of n elements.
    - Example: (1,1,1) is int[] A = {1,1,1}; in Java.
- So, a large vector {1,1,0,0,0,0,0,0,0,…. 10K elements} is
  - an array with 10K elements where only first two elements are 1s.

Is there a better way to represent this data?

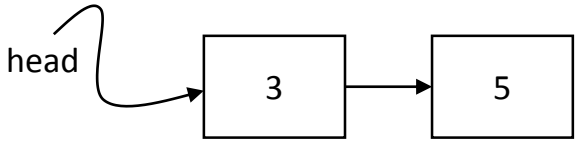
The Answer

- {1,1,0,0,0,0,0,0,0,…. 10K elements} is



A Linked List!

- {0,0,1,0,1,0,0,…..10K elements} is

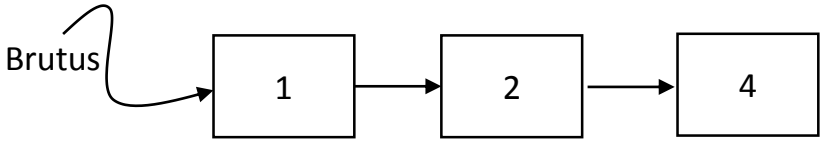


A Linked List!

Representing term-document Data

Documents

Terms	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0



Linked List Idea in Practice

Tokenization

- Task
  - Chop documents into pieces.
  - Throw away characters such as punctuations.
  - Remaining words are called tokens.
- Example
  - Document 1
    - I did enact Julius Caesar. I was killed i' the Capitol; Brutus killed me.
  - Document 2
    - So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Sort

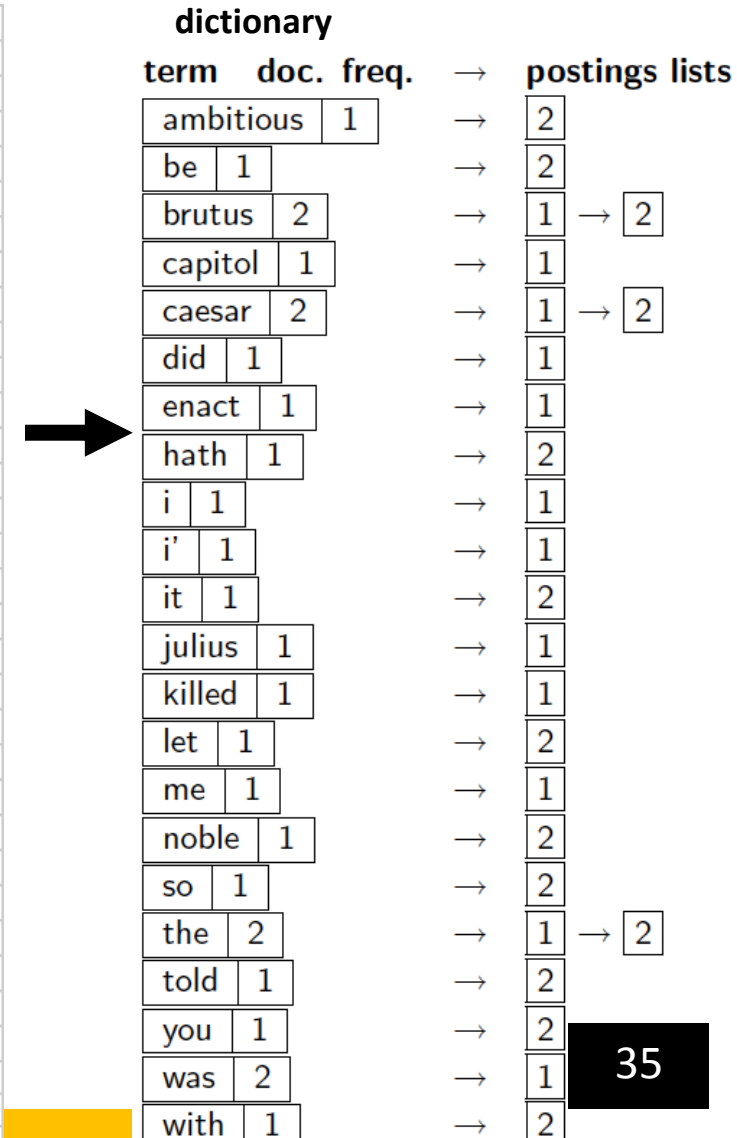
Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	

Inverted Index: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



Query Processing with Inverted Index

Boolean queries: Exact match

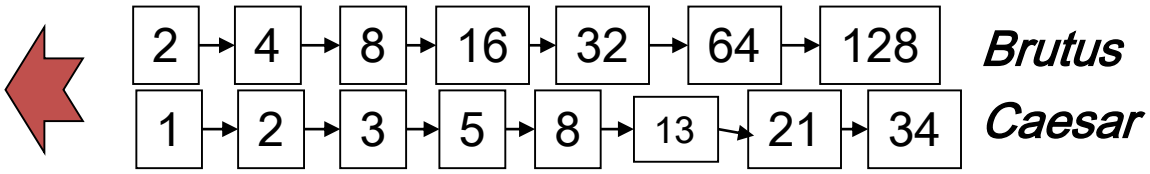
- The Boolean retrieval model is being able to ask a query that is a boolean expression:
  - Boolean queries are queries using *AND, OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
  - Perhaps the simplest model to build an IR system.

Query processing: AND

- Consider processing the query:

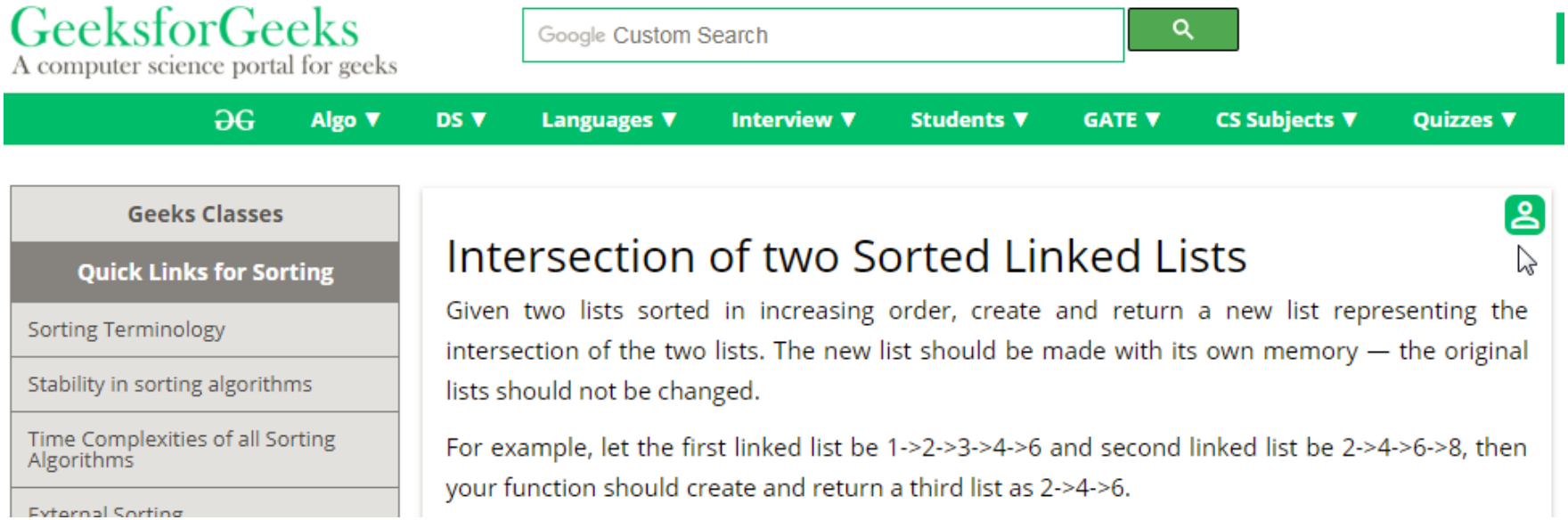
  Brutus *AND* ***Caesar***

  - Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.
  - Locate ***Caesar*** in the Dictionary;
    - Retrieve its postings.
  - "Merge" the two postings (intersect the document sets):



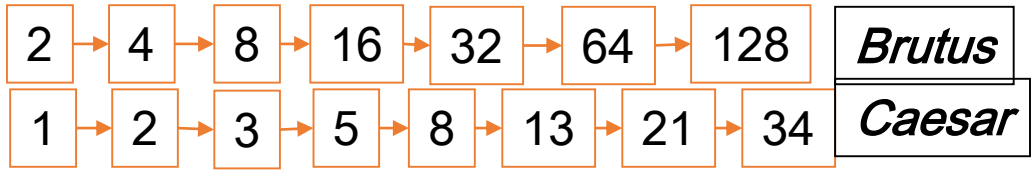
Common Interview Question

- https://www.geeksforgeeks.org/intersection-of-two-sorted-linked-lists/



The Merge

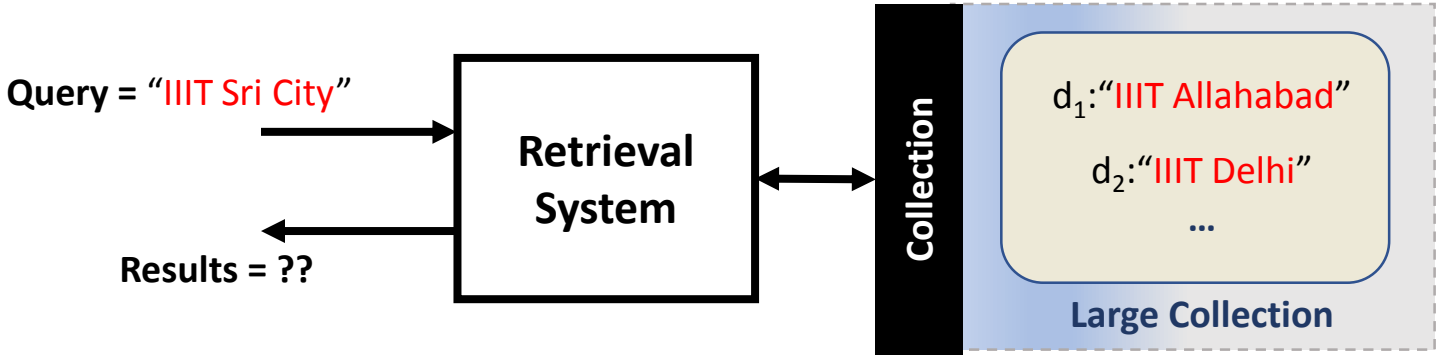
- Walk through the two postings simultaneously
  - Clue: Use two pointers



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.
Crucial: postings sorted by docID.

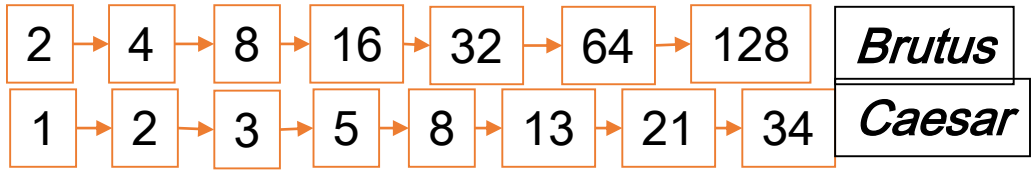
The Big Picture

- Content Processing
  - Build Term Document Matrix or Build Inverted Index
- Query Handling
  - Boolean AND or Intersect the Posting Lists (called merging process)



The Merge

- Walk through the two postings simultaneously
  - Clue: Use two pointers

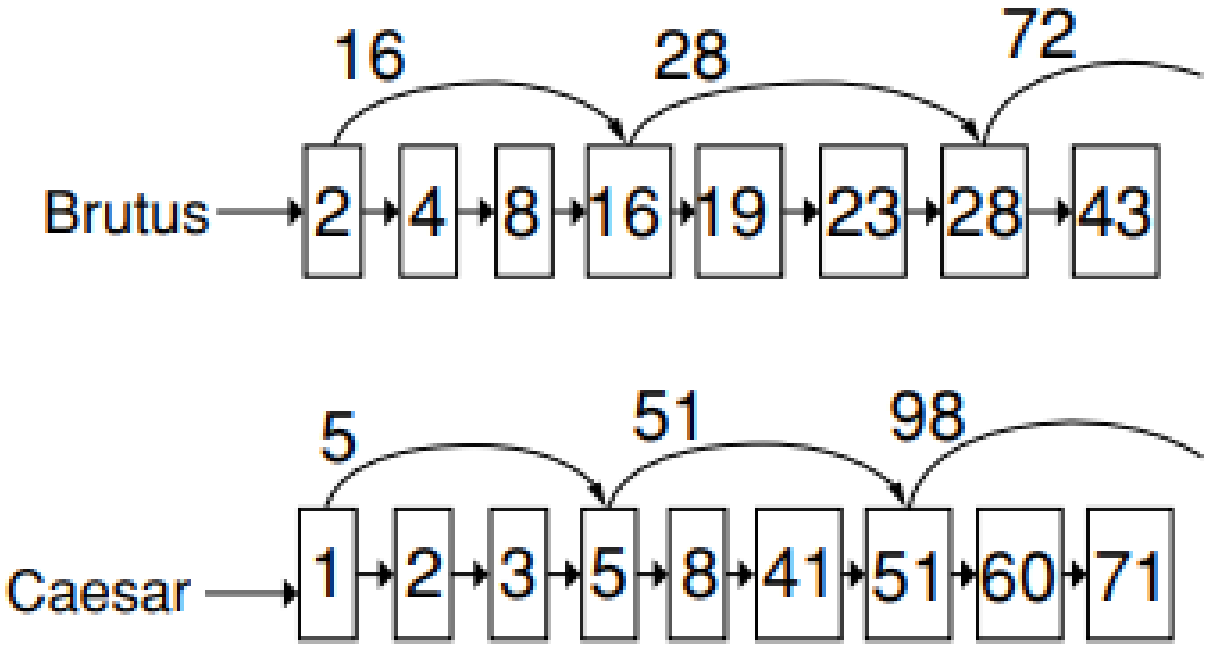


If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.
Crucial: postings sorted by docID.

Can we do better?

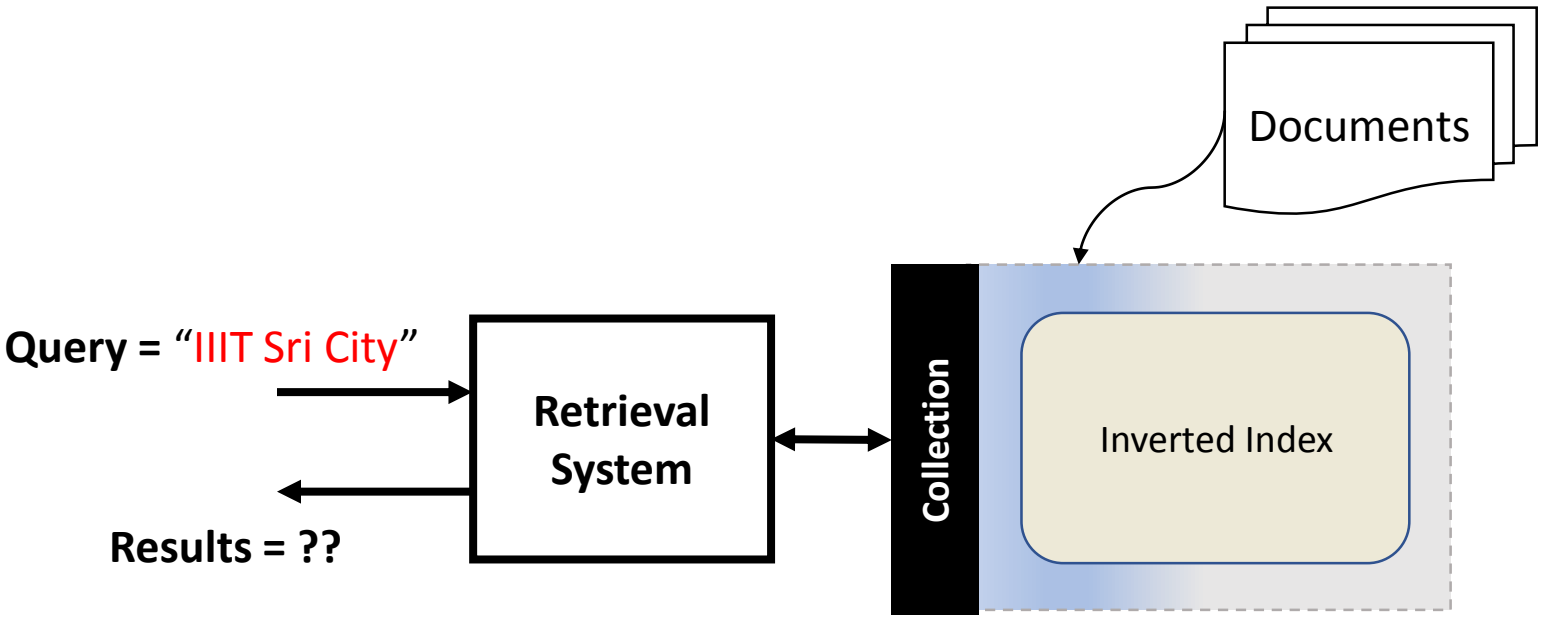
Inspired from multiple index idea of DBMS

Skip Pointers



Phrasal Queries

- What if we do not want to match “IIIT Delhi”?



One (bad) Approach

- Index all biwords
  - Friends, Romans, Countrymen → Friends Romans, Romans Countrymen
- How do you match the query IIIT Sri City, Chittoor?
  - **“IIIT Sri”** AND “**Sri City”** AND “**City Chittoor”** must exist.
- The Problem: “**IIIT**” AND “**Sri City**” AND “**Chittoor**” sounds like a much better query!
  - Natural Language Processing techniques can help in query formulation.

A Better Approach

- Store Positional Information

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

Extended Boolean Model with Positional Index

to, 993427:
⟨ 1, 6: ⟨7, 18, 33, 72, 86, 231⟩;
2, 5: ⟨1, 17, 74, 222, 255⟩;
4, 5: ⟨8, 16, 190, 429, 433⟩;
5, 2: ⟨363, 367⟩;
7, 3: ⟨13, 23, 191⟩; . . . ⟩

be, 178239:
⟨ 1, 2: ⟨17, 25⟩;
4, 5: ⟨17, 191, 291, 430, 434⟩;
5, 3: ⟨14, 19, 101⟩; . . . ⟩

"to" appears six times in d1 at positions 7, 18, ….
"to" appears 993K times overall.

Which document is likely to contain "to be"?

Proximity Search

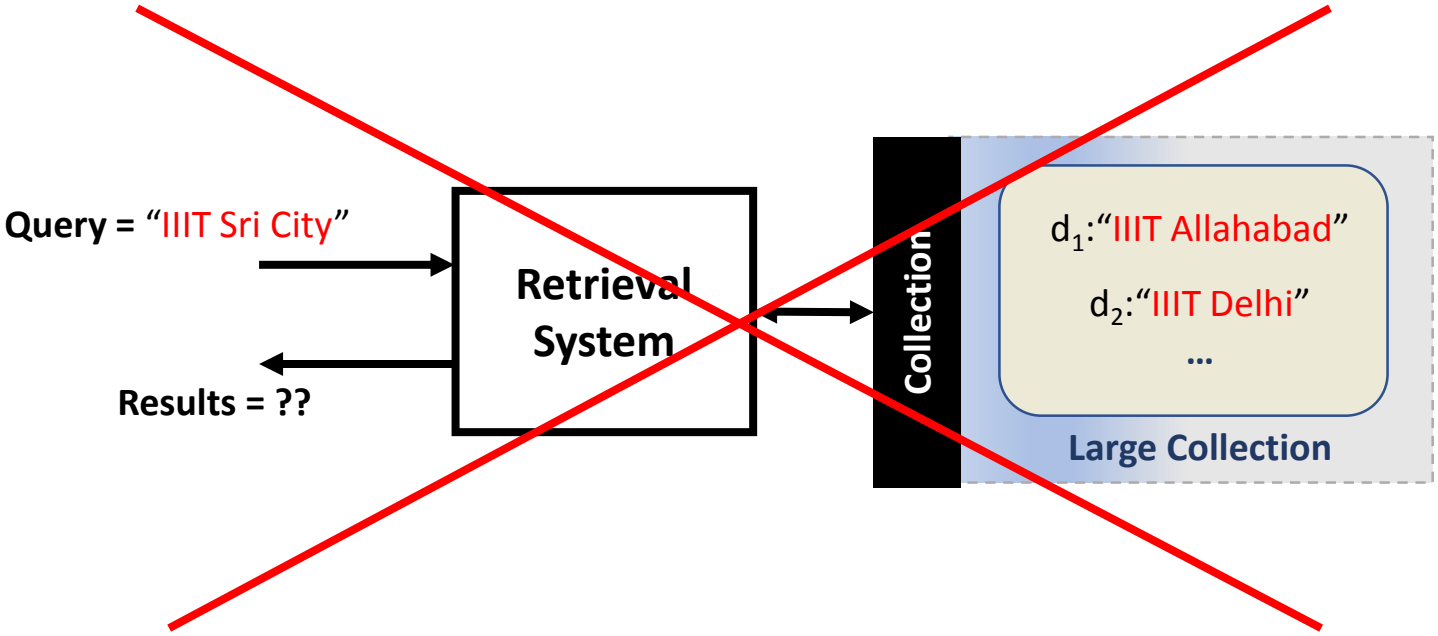
- IIIT /3 Chittoor
  - /k means “within k words of (on either side)”

- Merging postings is expensive
  - Index well-known phrases such as “Taj Mahal”

Combination Schemes

- biword index and positional index ideas can be combined.
- Use biword index or common phrases (such as Taj Mahal).
  - Avoids merging postings lists.
- Use positional index for other phrases (such as IIIT Chittoor).

The Big Picture



The Big Picture