# Information Retrieval
## An Overview

**Venkatesh Vinayakarao**
venkateshv@cmi.ac.in
http://vvtesh.co.in

Post-IOI Training Camp Workshop
Chennai Mathematical Institute

Text is the primary way that human knowledge is stored, and after speech, the primary way it is transmitted. -**Bill Frakes and Ricardo Baeza Yates.**

# Agenda

- What is Information Retrieval?
- How do search engines work?
  - Content Processing and Indexing
  - Ranking and Relevance
  - Evaluation

# Information

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

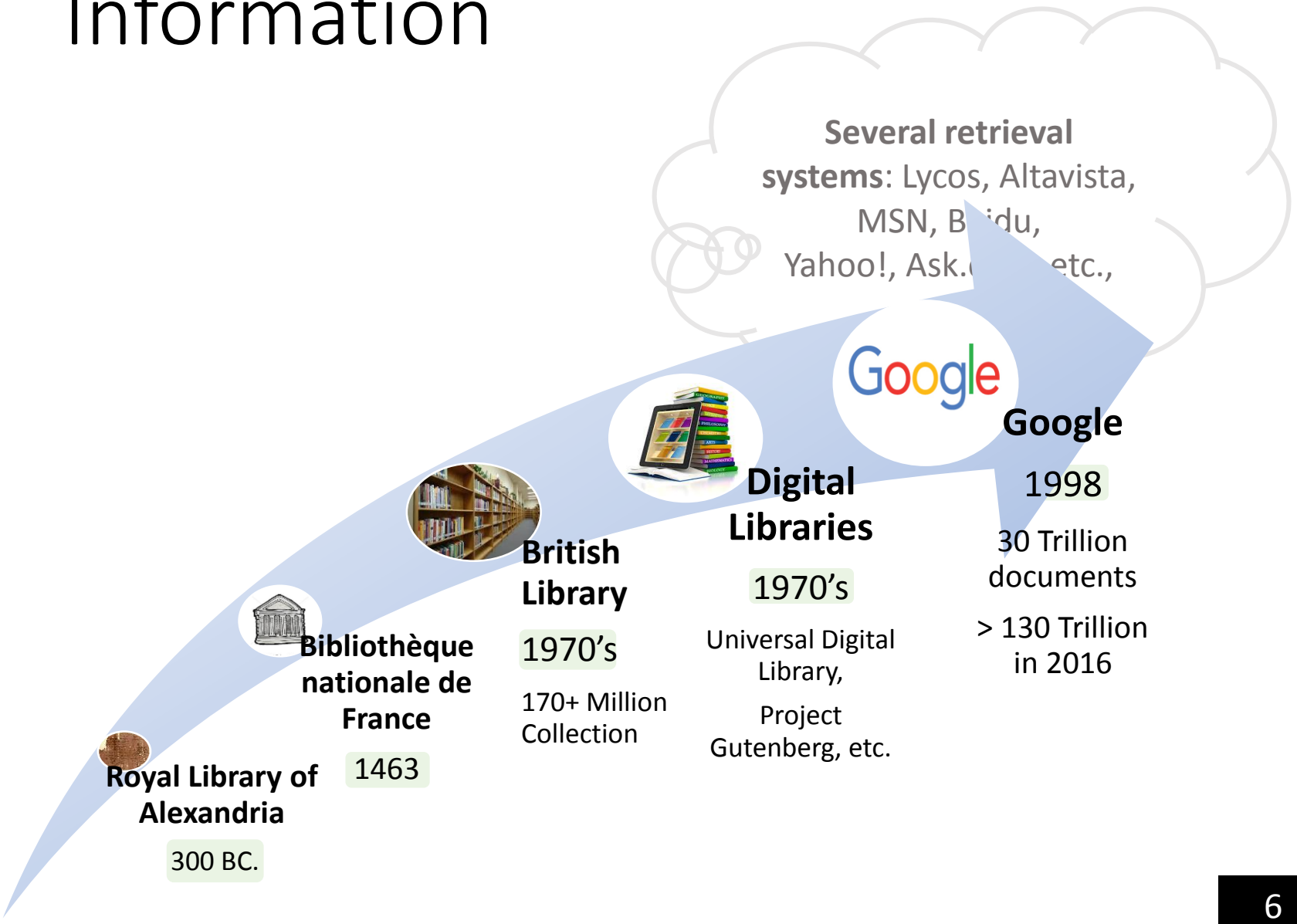
Information is any entity or form that provides the answer to a question of some kind or resolves uncertainty. – Wikipedia.

# Role of Information

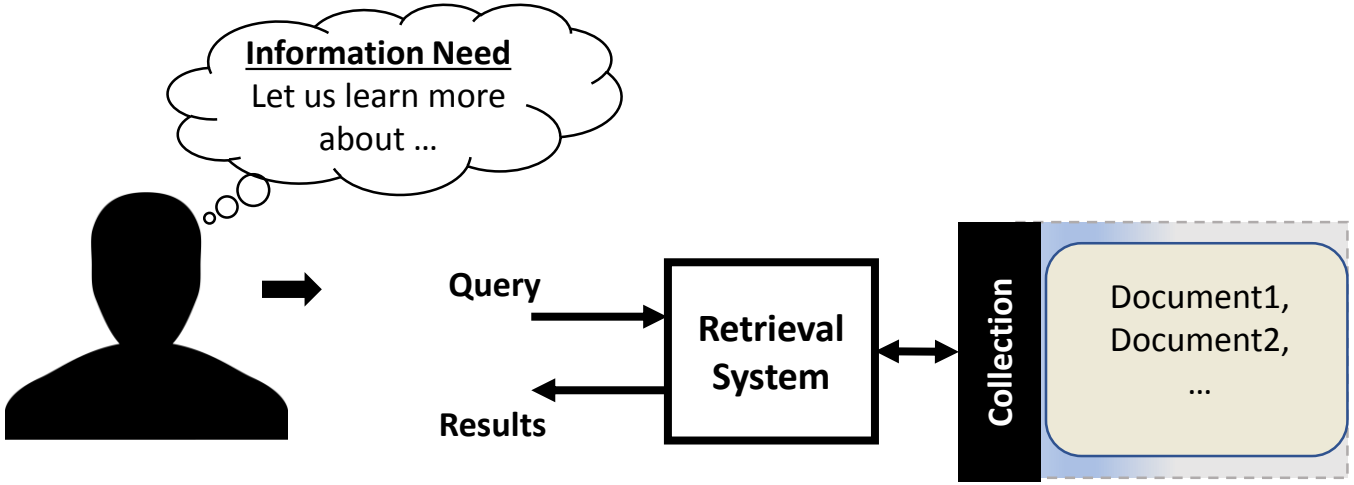
- If you only knew
  - Which stock to invest in?
  - What concepts to master for success in olympiad?
  - How to get into a top college?
  - Which course to register for?
  - What to study?
  - How to prepare for interviews?
  - …
- If only you had the information, you could rule this world!
- What happens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

# Solitary Confinement is Cruel

# Information



# What is Information Retrieval?



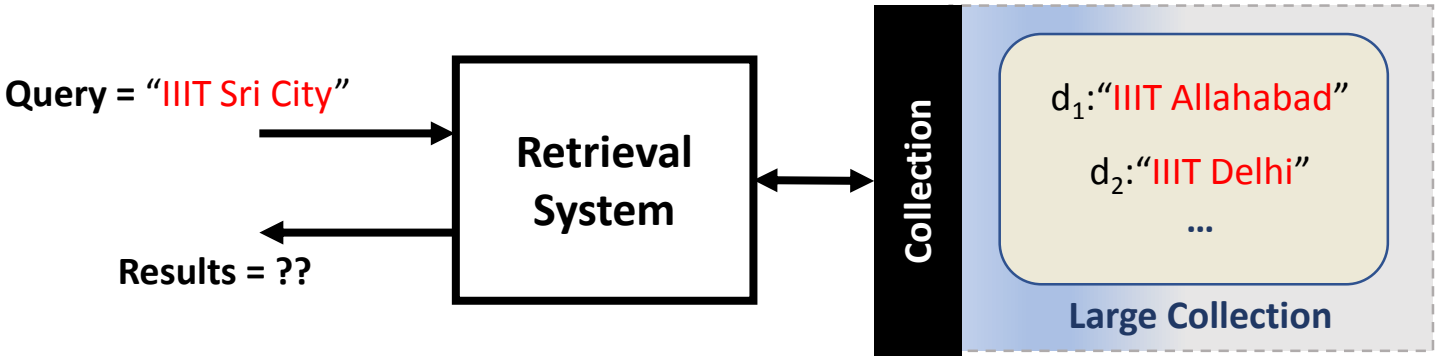

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an **information need** from within large collections.

– From the Manning et al. IR Book.

# 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: How to Build a Retrieval System?



# 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.

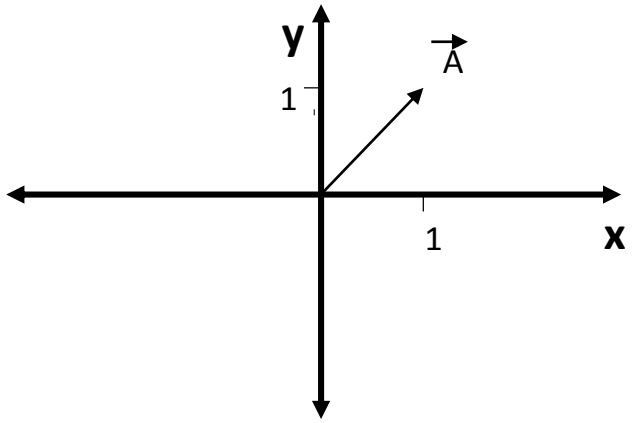
**Three iterations!**
**Quiz: Can we do better?**

# A Better Approach

**Revisiting**
**Linear Algebra**

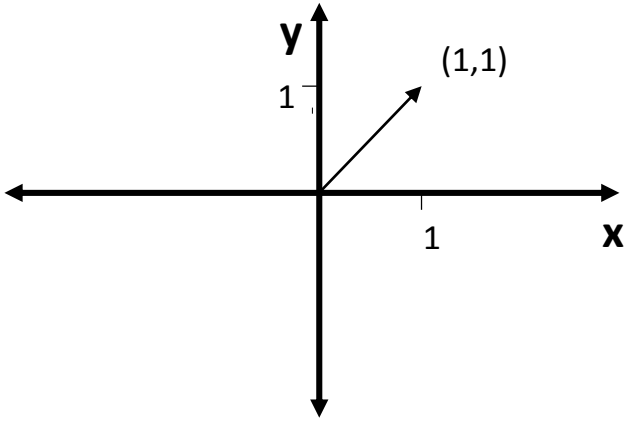
# Vectors

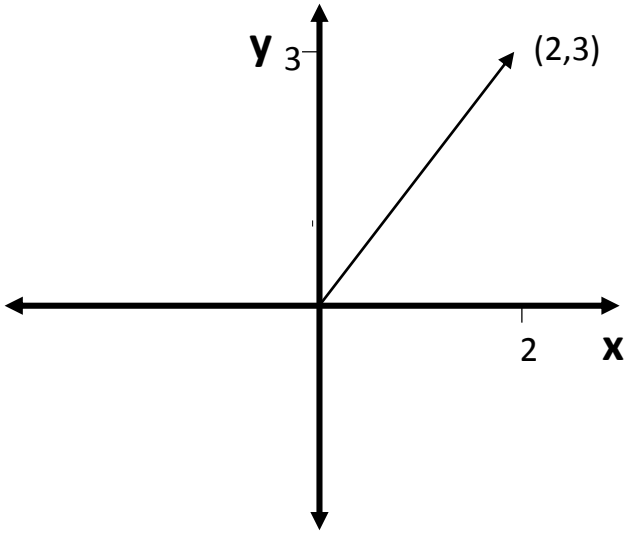
- Geometric entity which has magnitude and direction



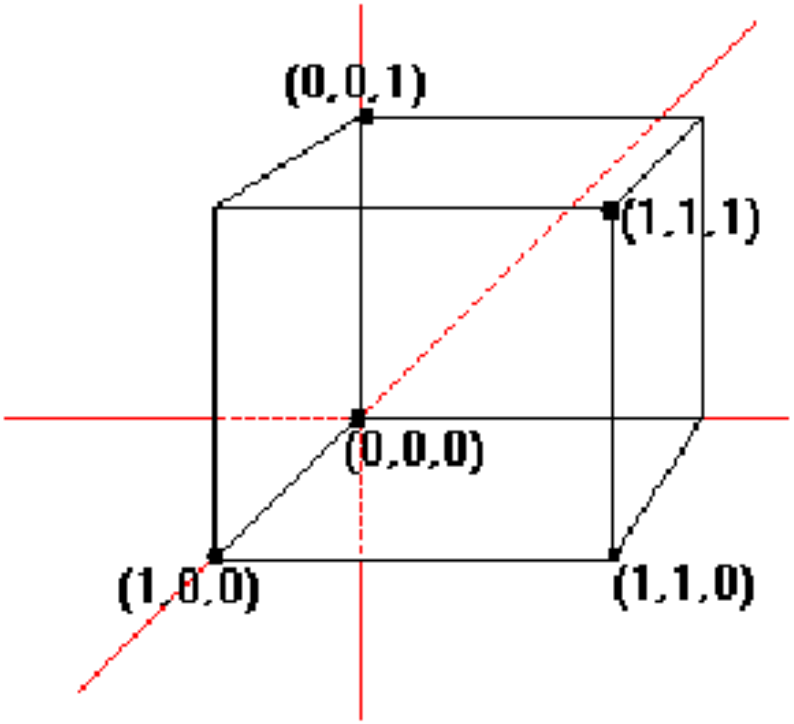

- If (x,y) is our vector of interest, this figure shows $\vec{A}$ vector = (1,1).

# How is (2,3) Different?



# What is (1,1,1) ?

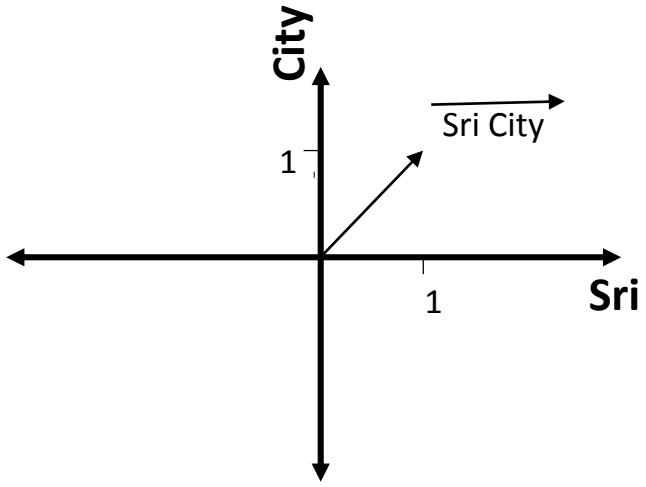


# Remember!

**A number is just a mathematical object. We give meaning to it!**

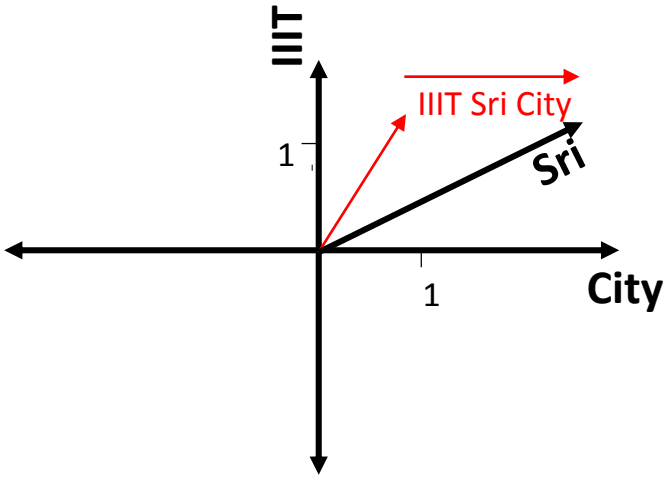
# Sentences are Vectors

- “Sri City” as a vector



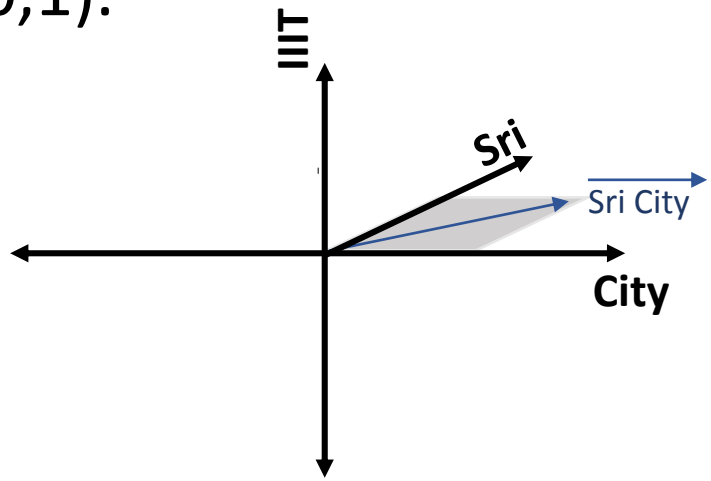
# Sentences are Vectors

- "IIIT Sri City" is a 3-dimensional vector



# Sentences are Vectors

- On this 3D space, “Sri City” vector will lie on the x (City) and z (Sri) plane. If (x,y,z) denotes the vector, “Sri City” is (1,0,1).



# More Linear Algebra…

- So, we learned to represent English phrases on the vector space.
- We need something more!

**Revisiting Matrices**

# Natural Language Phrases as Vectors

Let query q = "IIIT Sri City".

Let document, $d_1$ = "IIIT Sri City" and $d_2$ = "IIIT Delhi".

| | IIIT | Sri | City | Delhi |
|---|---|---|---|---|
| q | 1 | 1 | 1 | 0 |
| $d_1$ | 1 | 1 | 1 | 0 |
| $d_2$ | 1 | 0 | 0 | 1 |

q = (1,1,1,0), $d_1$= (1,1,1,0) and $d_2$ = (1,0,0,1)

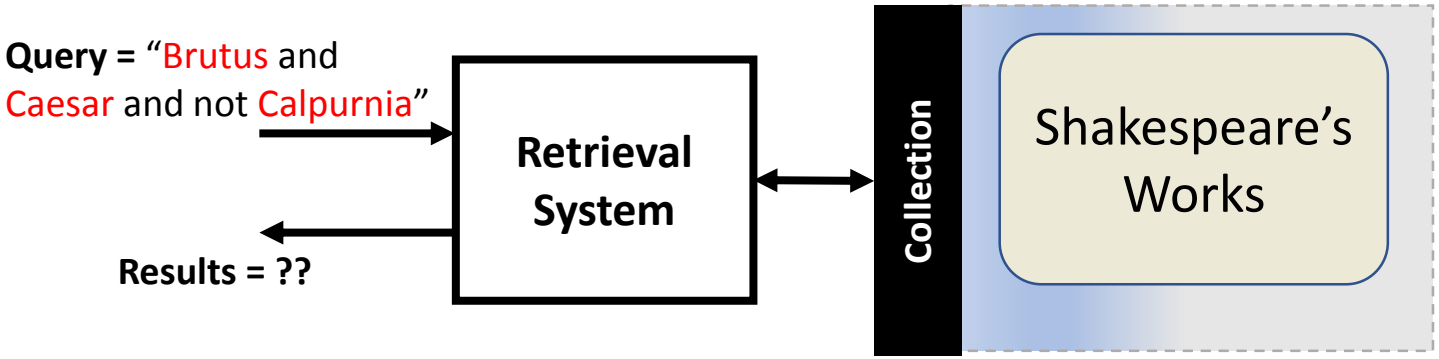
# Quiz

- Considering the following vectors:

| | IIIT | Sri | City | Delhi |
|---|---|---|---|---|
| q | 1 | 1 | 1 | 0 |
| $d_1$ | 1 | 1 | 1 | 0 |
| $d_2$ | 1 | 0 | 0 | 1 |

- What is the Natural Language (NL) equivalent of (0,1,1,0) ?
- What is the NL equivalent of (1,0,0,1) ?
- What is the vector for Delhi?
- What is the NL equivalent of q here?

# Boolean Operators in Queries



# A term-document Matrix Example

**Documents**

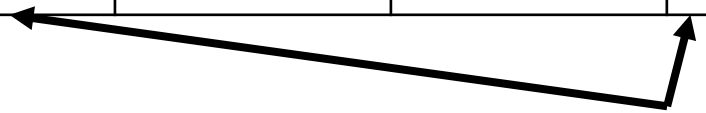
| Terms | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0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

"Brutus and Caesar and not Calpurnia"

# Answering Boolean Queries

"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:
  - More distinct words are added to the matrix i.e., 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?

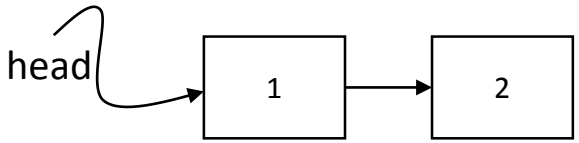
# The Problem

- 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 sparse data?**

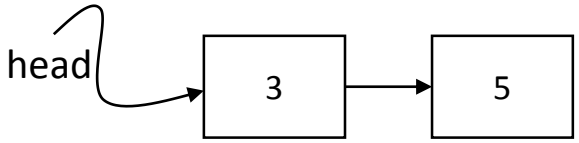
# The Answer

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




**A Linked List!**

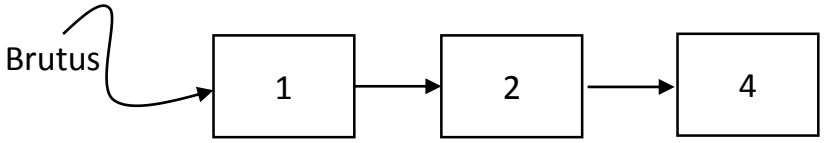
- {0,0,1,0,1,0,0,…..10K elements} is



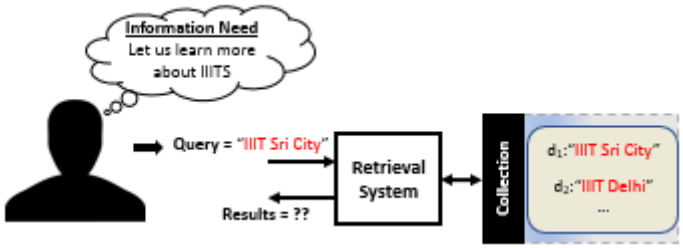

**A Linked List!**

# Representing term-document Data

| Terms | Documents | | | | | |
|---|---|---|---|---|---|---|
| | 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 |



# Review




## 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.

Documents

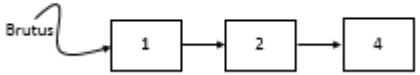
| Terms | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0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

"Brutus and Caesar and not Calpurnia"

| | | | | | |
|---|---|---|---|---|---|
| 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.

~~int[] A = {1,1,1};~~



# Content Processing & Indexing

Tokenization, Stop Word Removal, Sorting, Dictionary & Postings

# Tokenization

- Task
  - Chop documents into words.
  - Throw away characters such as punctuations.
  - Remaining words are called tokens.
  - Drop uninteresting tokens (stop words)
  - Remaining words are called terms.
- 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

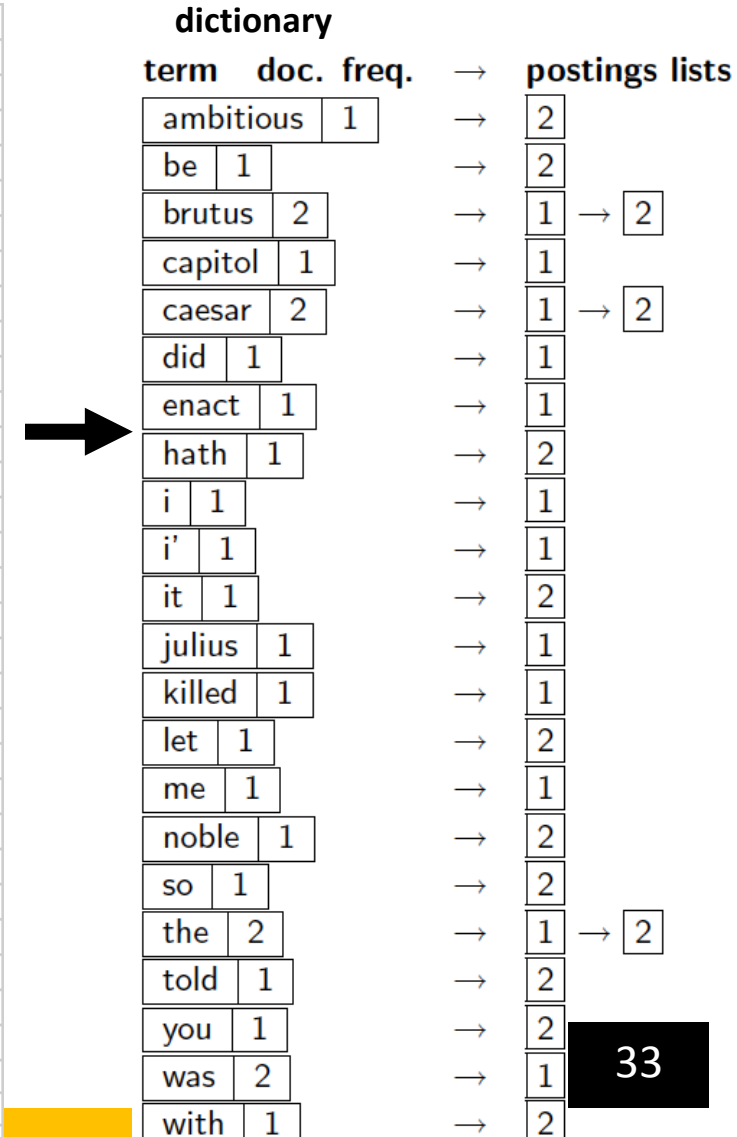
| Term | docID |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 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 |



# A Case of Exact match



- The Boolean retrieval model
  - Match or No-Match!
- Boolean Expressions in Queries
  - Queries using *AND, OR* and *NOT* to join query terms.
    - Views each document as a set of words.
- Just a simple IR system.

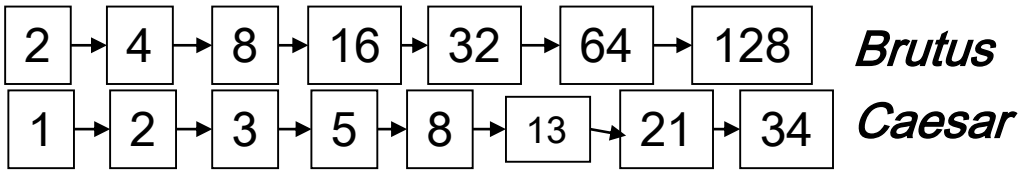
# Query processing: AND



- Consider processing the query:

  ***Brutus*** *AND* ***Caesar***

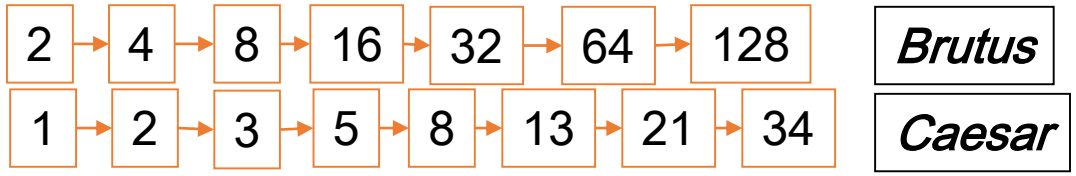
  - Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.
  - Locate ***Caesar*** in the Dictionary;
    - Retrieve its postings.
  - "Merge" the two postings (intersect the document sets):



# The Merge



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

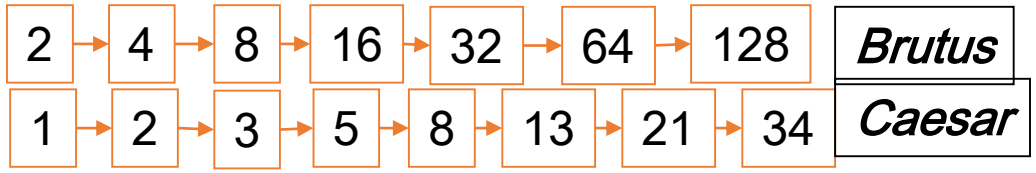



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.
Crucial: postings sorted by docID.

# The Merge

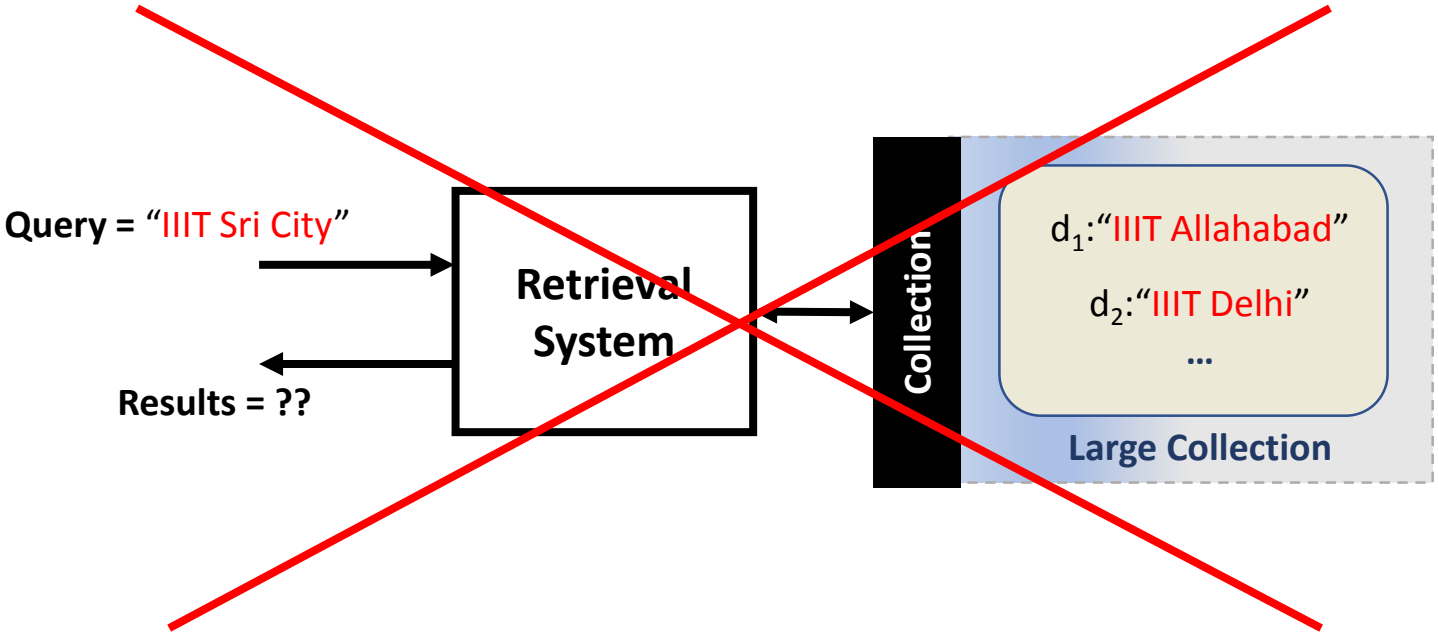


- Walk through the two postings simultaneously
  - Clue: Use two pointers

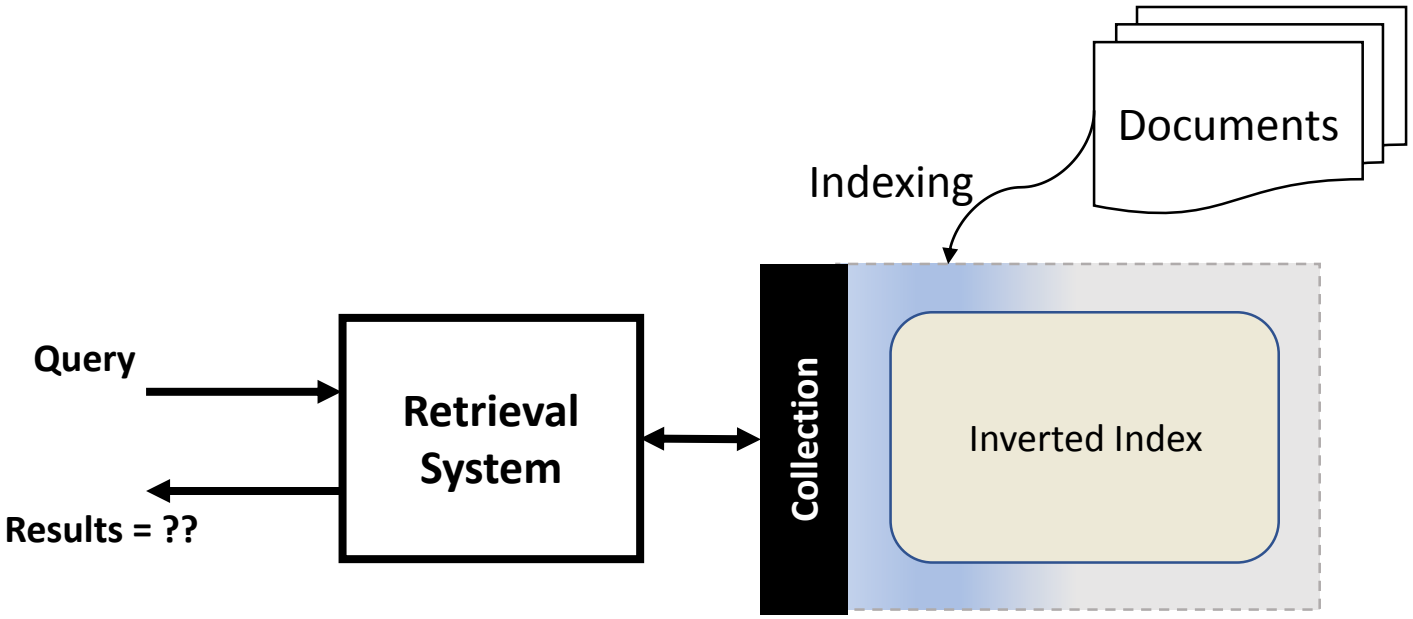



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.
Crucial: postings sorted by docID.

# The Big Picture



# The Big Picture



# Relevance and Ranking

# Similarity Score

- D1 = “Chennai”
- D2 = “Delhi”

- Quiz
  - What is the angle between D1 and D2 vectors?
  - On a scale of 0 – 1, how similar are D1 and D2?

# 0 – 90 to 1 – 0: How?

| | 0° | 30° | 45° | 60° | 90° |
|---|---|---|---|---|---|
| sin $\theta$ | 0 | $\frac{1}{2}$ | $\frac{1}{\sqrt{2}}$ | $\frac{\sqrt{3}}{2}$ | 1 |
| cos $\theta$ | 1 | $\frac{\sqrt{3}}{2}$ | $\frac{1}{\sqrt{2}}$ | $\frac{1}{2}$ | 0 |
| tan $\theta$ | 0 | $\frac{1}{\sqrt{3}}$ | 1 | $\sqrt{3}$ | Not defined |

# Back to Trigonometry: Dot Product

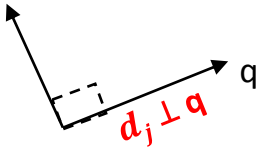
- If x and y are non-unit vectors, what is the cosine of angle between them (cos Θ)?

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos(\theta)$$

# Matching Documents to Queries

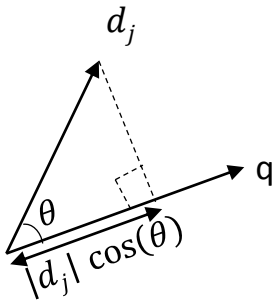
- Document as a vector of term-occurrence

$$d_j = (w_{1j}, w_{2j}, \ldots w_{nj})$$

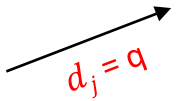



- Query as a vector of term-occurrence

$$q = (w_{1q}, w_{2q}, \ldots wmq)$$




- Similarity between these vectors can be represented as




$$Cosine\ Similarity = \cos(\theta) = \frac{d_j . q}{||d_j||\ ||q||}$$

# Example

Let query q = "BITS Pilani".

Let document, $d_1$ = "BITS Pilani Goa Campus" and $d_2$ = "IIIT Delhi".

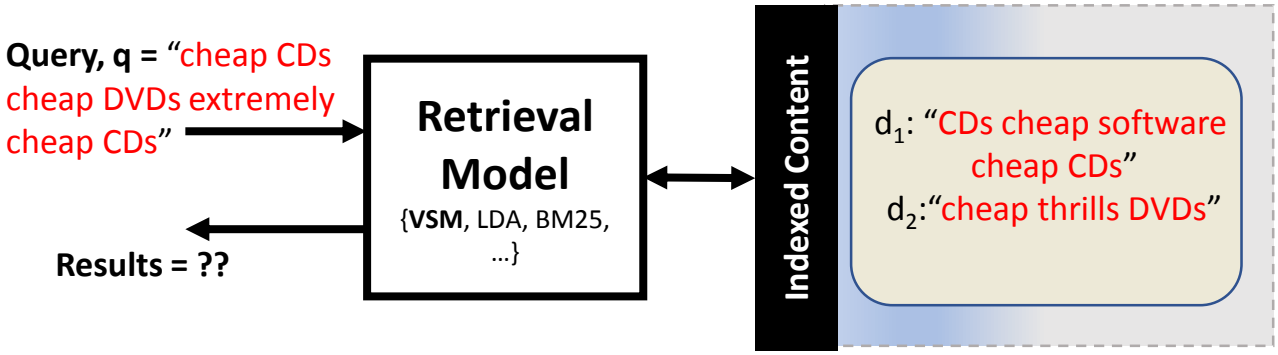
| | BITS | Pilani | Goa | Campus | IIIT | Delhi |
|---|---|---|---|---|---|---|
| q | 1 | 1 | 0 | 0 | 0 | 0 |
| $d_1$ | 1 | 1 | 1 | 1 | 0 | 0 |
| $d_2$ | 0 | 0 | 0 | 0 | 1 | 1 |

In our VSM, q = (1,1,0,0,0,0), $d_1$= (1,1,1,1,0,0) and $d_2$ = (0,0,0,0,1,1)

**similarity($d_1$, q)** = $\frac{d_1.q}{||d_1||\,||q||} = \frac{1.1 + 1.1}{\sqrt{1^2+1^2+1^2+1^2}\,\sqrt{1^2+1^2}} = 0.71.$

**similarity($d_2$, q)** = $\frac{d_2.q}{||d_2||\,||q||} = 0.$

# Which Document to Retrieve?




| | cheap | CDs | DVDs | extremely | software | thrills |
|---|---|---|---|---|---|---|
| q | 3 | 2 | 1 | 1 | 0 | 0 |
| $d_1$ | 2 | 2 | 0 | 0 | 1 | 0 |
| $d_2$ | 1 | 0 | 1 | 0 | 0 | 1 |

$sim(q,d_1) = 0.86$

$sim(q,d_2) = 0.59$

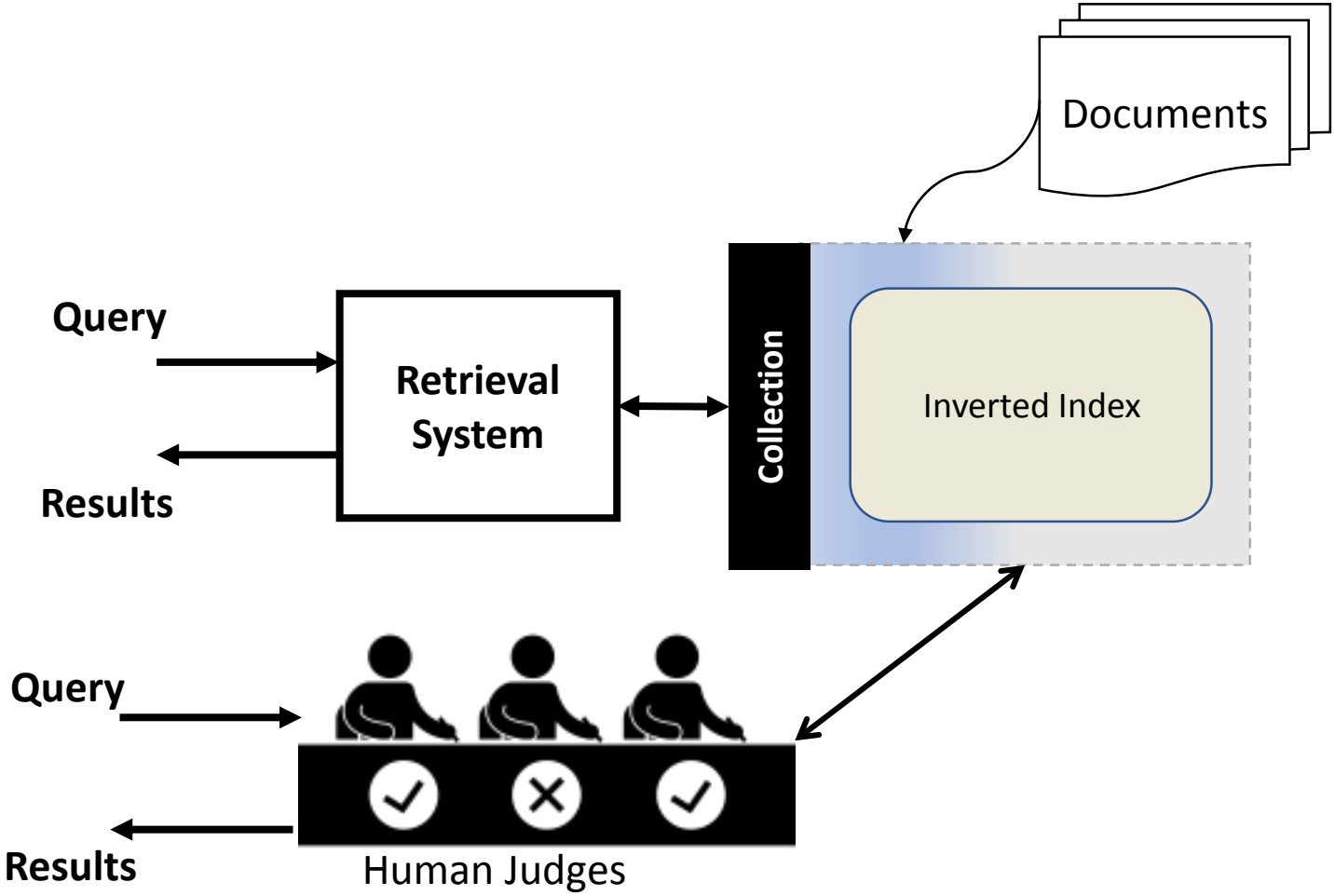
# Evaluation

Comparing Retrieval Systems

# How Good is Our System?

- A **collection** having the following contents
  - d1: IIIT ALLAHABAD
  - d2: IIIT DELHI
  - d3: IIIT GUWAHATI
  - d4: CMI
  - d5: IIIT SRI CITY
  - d6: KREA SRI CITY
- **Query** is
  - SRI CITY
- **Result** is
  - IIIT SRI CITY
  - KREA SRI CITY

# Evaluation


Documents
Query
Retrieval System
Results
Collection
Inverted Index
Query
Results
Human Judges

# How Good is Our System?

- A **collection** having the following contents
  - d1: IIIT ALLAHABAD
  - d2: IIIT DELHI
  - d3: IIIT GUWAHATI
  - d4: CMI
  - d5: IIIT SRI CITY
  - d6: KREA SRI CITY
- **Query** is
  - IIIT
- **Result** is
  - IIIT SRI CITY
  - KREA SRI CITY

# Objective

We want all relevant documents and only relevant documents

# Relevance

- How many **relevant** documents?
  - Four (IIIT SRI CITY, IIIT ALLAHABAD, IIIT DELHI, IIIT GUWAHATI)
- How many **retrieved** documents?
  - Two (IIIT SRI CITY, KREA SRI CITY)

How to quantify the “goodness” of our system?

# Terminology

- Documents we see in results are "**positive**"
  - Positive
    - + IIIT SRI CITY,
    - + KREA SRI CITY
  - Negative
    - - IIIT ALLAHABAD
    - - IIIT DELHI
    - - IIIT GUWAHATI
    - - CMI

# Terminology

- Documents that we correctly classify are "**true**"
  - Positive
    - + IIIT SRI CITY (true)
    - + KREA SRI CITY
  - Negative
    - - IIIT ALLAHABAD
    - - IIIT DELHI
    - - IIIT GUWAHATI
    - - CMI (true)

**Here, query is "IIIT"**

# Quiz

- All retrieved results =
  1. tp + fp
  2. tp + fn
  3. tn + fp
  4. tn + fn

**Legend**

tp = true positive

tn = true negative

fp = false positive

fn = false negative

# Quiz

- All retrieved results =
  1. tp + fp
  2. tp + fn
  3. tn + fp
  4. tn + fn

**Legend**

tp = true positive

tn = true negative

fp = false positive

fn = false negative

# Quiz

- All relevant results =
  1. tp + fp
  2. tp + fn
  3. tn + fp
  4. tn + fn

**Legend**

tp = true positive

tn = true negative

fp = false positive

fn = false negative

# Quiz

- All relevant results =
  1. tp + fp
  2. tp + fn
  3. tn + fp
  4. tn + fn

**Legend**

tp = true positive

tn = true negative

fp = false positive

fn = false negative

# You have 100% Precision

- Everything you retrieved were relevant.
  - fp = 0

# You have 100% Recall when

- You retrieved everything that were relevant. (Note: You could have retrieved more).
  - fn = 0

# Example

- The following list of Rs and Ns represents relevant (R) and nonrelevant (N) returned documents in a list of 20 documents retrieved in response to a query from a collection of 10,000 documents. This list shows 6 relevant documents. Assume that there are 8 relevant documents in total in the collection. Calculate Precision and Recall.

RRNNN NNNRN RNNNR NNNNR

# Precision and Recall

- Precision = 6/20
- Recall = 6/8

# Precision and Recall

**Precision**: fraction of retrieved docs that are relevant

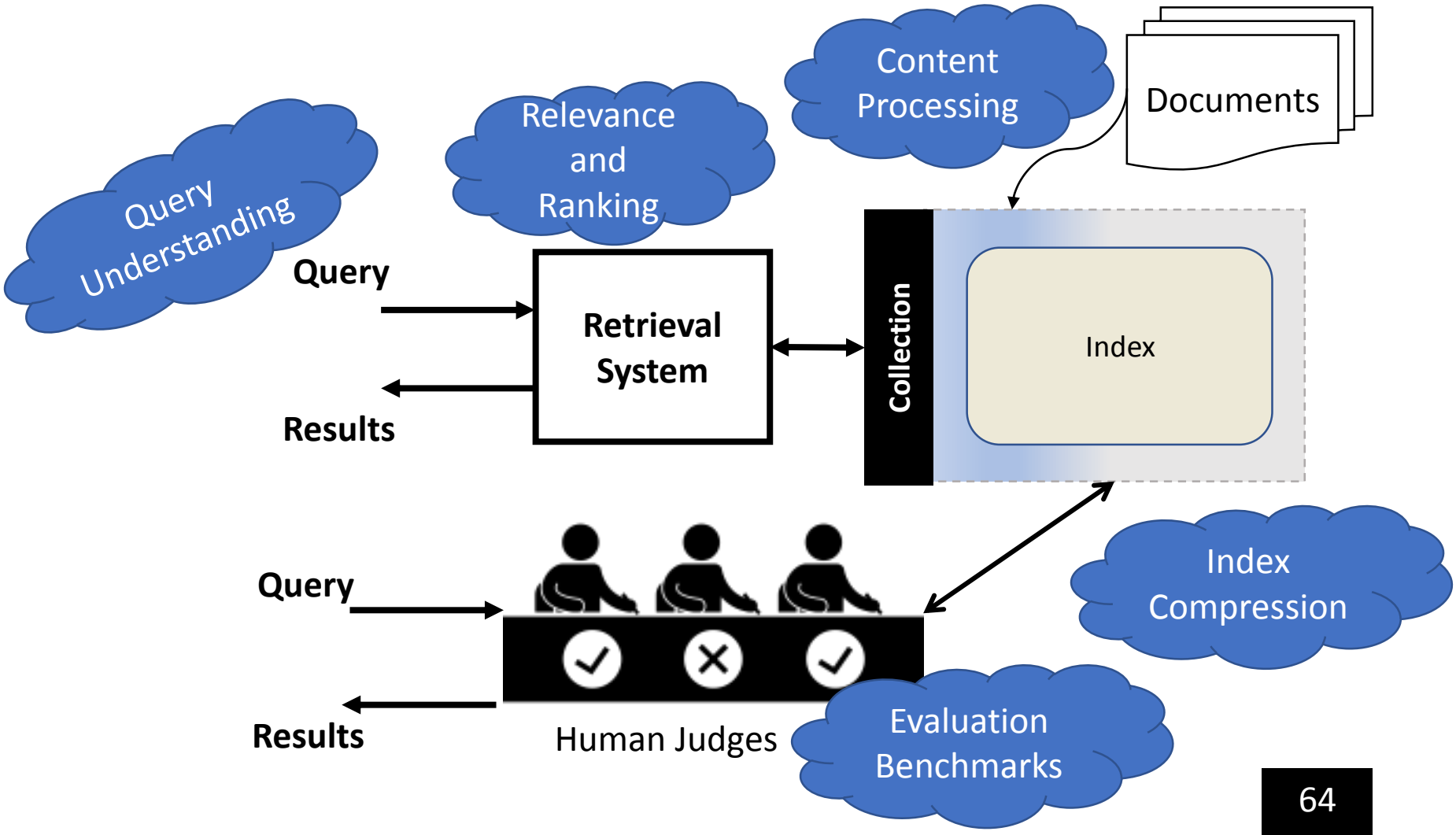
$$= tp/(tp + fp)$$

**Recall**: fraction of relevant docs that are retrieved

$$= tp/(tp + fn)$$

| | Relevant | Nonrelevant |
|---|---|---|
| Retrieved | tp | fp |
| Not Retrieved | fn | tn |

# Information Retrieval – Road Ahead



# Thank You

*If you're changing the world, you're working on important things. You're excited to get up in the morning. – Larry Page.*