



**Monsoon 2019**

**Creating Inverted Index**

# **I n f o r m a t i o n R e t r i e v a l**

**by**

**Dr. Rajendra Prasath**

**Indian Institute of Information Technology**

**Sri City – 517 646, Andhra Pradesh, India**

# Information Retrieval?



# Recap: Text Collections

## ✧ **Structured data**

- ✧ Information stored DB
- ✧ Strict format
- ✧ Limitation
- ✧ Not all data collected is structured

## ✧ **Semi-structured data**

- ✧ Data may have certain structure but not all information collected has identical structure
- ✧ Some attributes may exist in some of the entities of a particular type but not in others

## ✧ **Unstructured data**

- ✧ Very limited indication of data type
- ✧ For example, look into a simple text document

# Look at 3 documents

- d<sub>1</sub>- Darjeeling** is a city and a municipality in the Indian state of West Bengal. It is located in the Lesser Himalayas at an elevation of 6,700 feet
- d<sub>2</sub>- Darjeeling** is noted for its tea industry, its views of Kangchenjunga, the world's third-highest mountain, and the **Darjeeling** Himalayan Railway, a UNESCO World Heritage Site
- d<sub>3</sub>- Darjeeling** is the headquarters of the **Darjeeling** District which has a partially autonomous status within the state of West Bengal. It is also a tourist destination in India

# Unique words and Counts?

$d_1$

2 the  
2 of  
2 is  
2 in  
2 a  
1 state  
1 municipality  
1 located  
1 feet  
1 elevation  
1 city  
1 at  
1 and  
1 an  
1 West  
1 Lesser  
1 It  
1 Indian  
1 Himalayas  
1 Darjeeling  
1 Bengal  
1 6,700

$d_2$

2 the  
2 its  
2 Darjeeling  
1 world's  
1 views  
1 third-highest  
1 tea  
1 of  
1 noted  
1 mountain  
1 is  
1 industry,  
1 for  
1 and  
1 a  
1 World  
1 UNESCO  
1 Site  
1 Railway  
1 Kangchenjunga  
1 Himalayan  
1 Heritag

$d_3$

3 the  
2 of  
2 is  
2 a  
2 Darjeeling  
1 within  
1 which  
1 tourist  
1 status  
1 state  
1 partially  
1 in  
1 headquarters  
1 has  
1 destination  
1 autonomous  
1 also  
1 West  
1 It  
1 India  
1 District  
1 Bengal



# Documents – Words / Terms\*

✧ How to construct Terms - documents

Doc ID	Terms	# Words
$d_1$	6,700 (1), Bengal. (1), Darjeeling (1), Himalayas (1), Indian (1), It (1), Lesser (1), West (1), a (2), an (1), and (1), at (1), city (1), elevation (1), feet (1), in (2), is (2), located (1), municipality (1), of (2), state (1), the (2),	22
$d_2$	Darjeeling (2), Heritage (1), Himalayan (1), Kangchenjunga, (1), Railway, (1), Site (1), UNESCO (1), World (1), a (1), and (1), for (1), industry, (1), is (1), its (2), mountain, (1), noted (1), of (1), tea (1), the (2), third-highest (1), views (1), world's (1),	22
$d_3$	Bengal. (1), Darjeeling (2), District (1), India (1), It (1), West (1), a (2), also (1), autonomous (1), destination (1), has (1), headquarters (1), in (1), is (2), of (2), partially (1), state (1), status (1), the (3), tourist (1), which (1), within (1),	22

NOTE: “**Words**” and “**Terms**” are interchangeably used throughout the course

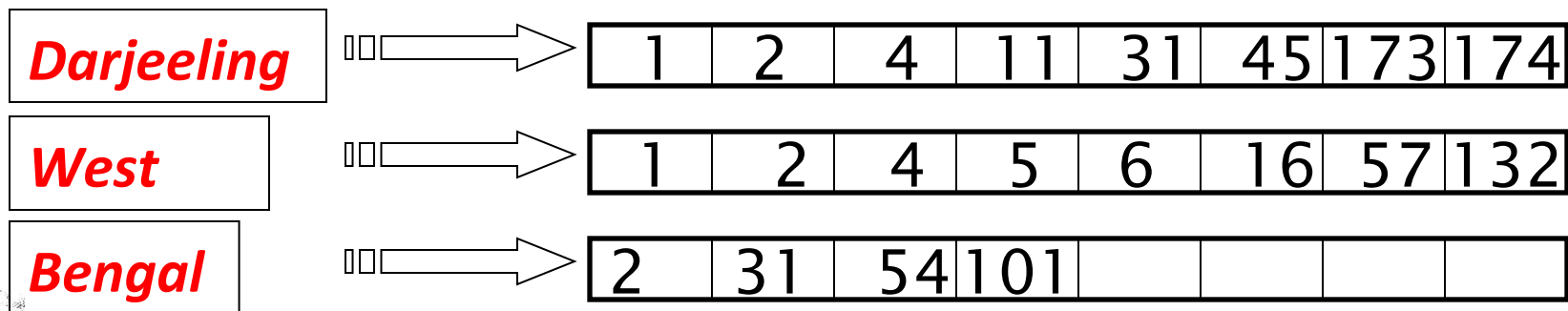
# Terms - Documents

Terms	$d_1$	$d_2$	$d_3$	...	$d_n$
the	2	2	3	...	0
a	2	1	2	...	1
Darjeeling	1	2	2	...	0
is	2	1	2	...	0
of	2	1	2	...	0
in	2	0	0	...	1
and	1	1	0	...	0
Bengal	1	0	1	...	0
It	1	0	1	...	0
Its	0	2	0	...	2
state	1	0	1	...	0
West	1	0	1	...	1

NOTE: “**Words**” and “**Terms**” are interchangeably used throughout the course

# Inverted index

- ✧ For each term  $t$ , we must store a list of all documents that contain  $t$ .
  - ✧ Identify each doc by a docID, a document serial number
- ✧ Can we use fixed-size arrays for this?

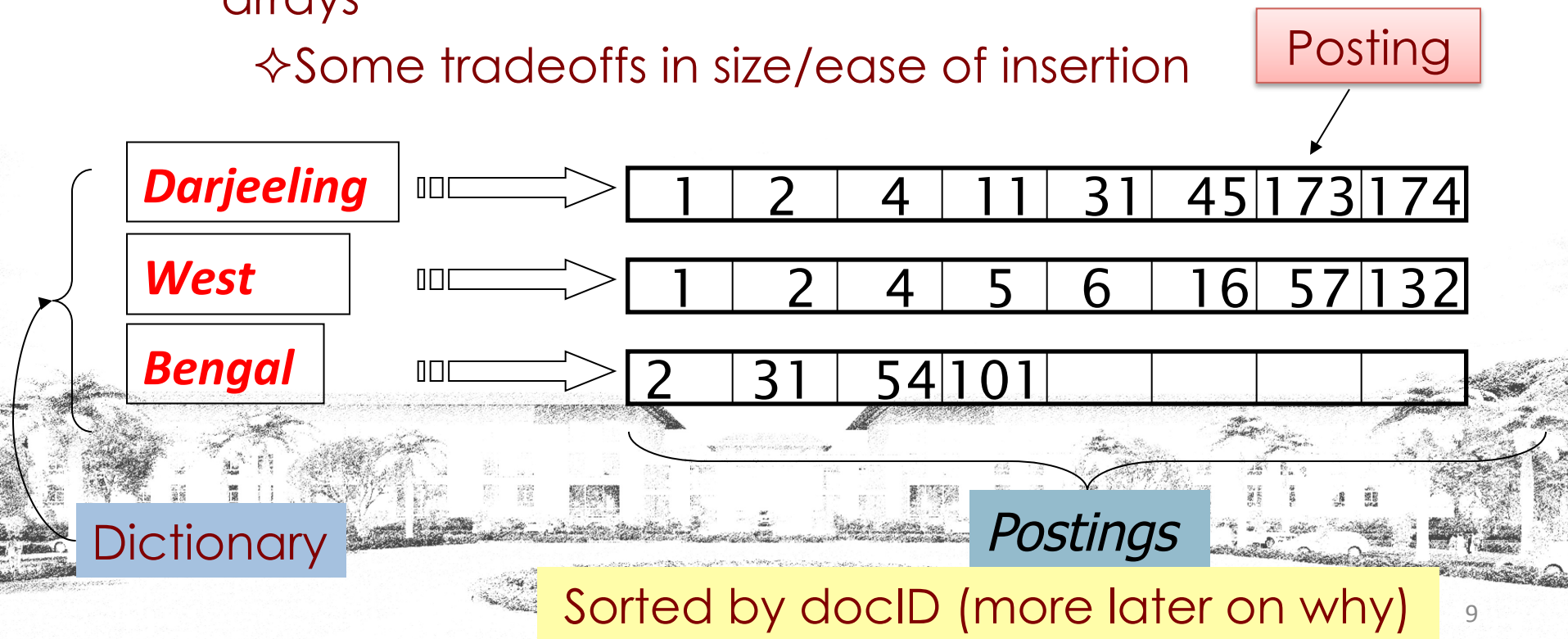


What happens if the word **Darjeeling** is added to document 14?



# Inverted index

- ✧ We need variable-size postings lists
  - ✧ On disk, a continuous run of postings is normal and best
  - ✧ In memory, can use linked lists or variable length arrays
  - ✧ Some tradeoffs in size/ease of insertion



# Inverted index construction

Documents to  
be indexed



Friends, Romans, countrymen.  
⋮

Tokenizer

Token stream

Friends

Romans

Countrymen

Linguistic modules

Modified tokens

friend

roman

countryman

Indexer

*friend*

*roman*

*countryman*

2

4

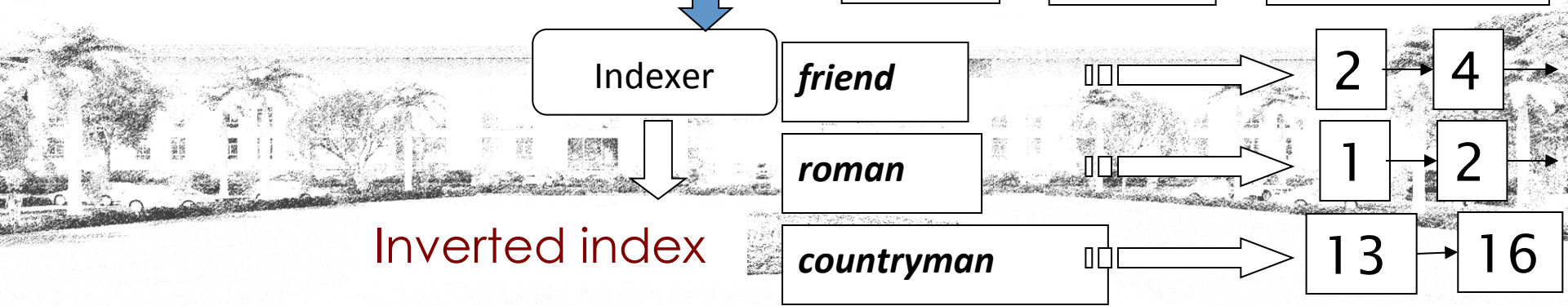
1

2

13

16

Inverted index



# Initial stages of text processing

## ✧ **Tokenization**

- ✧ Cut character sequence into word tokens
  - ✧ Deal with “John’ s”, a state-of-the-art solution

## ✧ **Normalization**

- ✧ Map text and query term to same form
  - ✧ You want U.S.A. and USA to match

## ✧ **Stemming**

- ✧ We may wish different forms of a root to match
  - ✧ authorize, authorization

## ✧ **Stop words**

- ✧ We may omit very common words (or not)
  - ✧ the, a, to, of

# Indexer steps:

## Token sequence

✧ Sequence of  
(Modified token, Document ID) pairs

Doc 1

I did enact Julius  
Caesar I was killed  
i' the Capitol;  
Brutus killed me.

Doc 2

So let it be with  
Caesar. The noble  
Brutus hath told you  
Caesar was ambitious

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

# Indexer steps: Sort

- Sort by terms
  - And then docID

**Core indexing step**

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	2



# Indexer steps: Dictionary & Postings

- ✧ Multiple term entries in a single document are merged.
- ✧ Split into Dictionary and Postings
- ✧ Doc. frequency information is added.

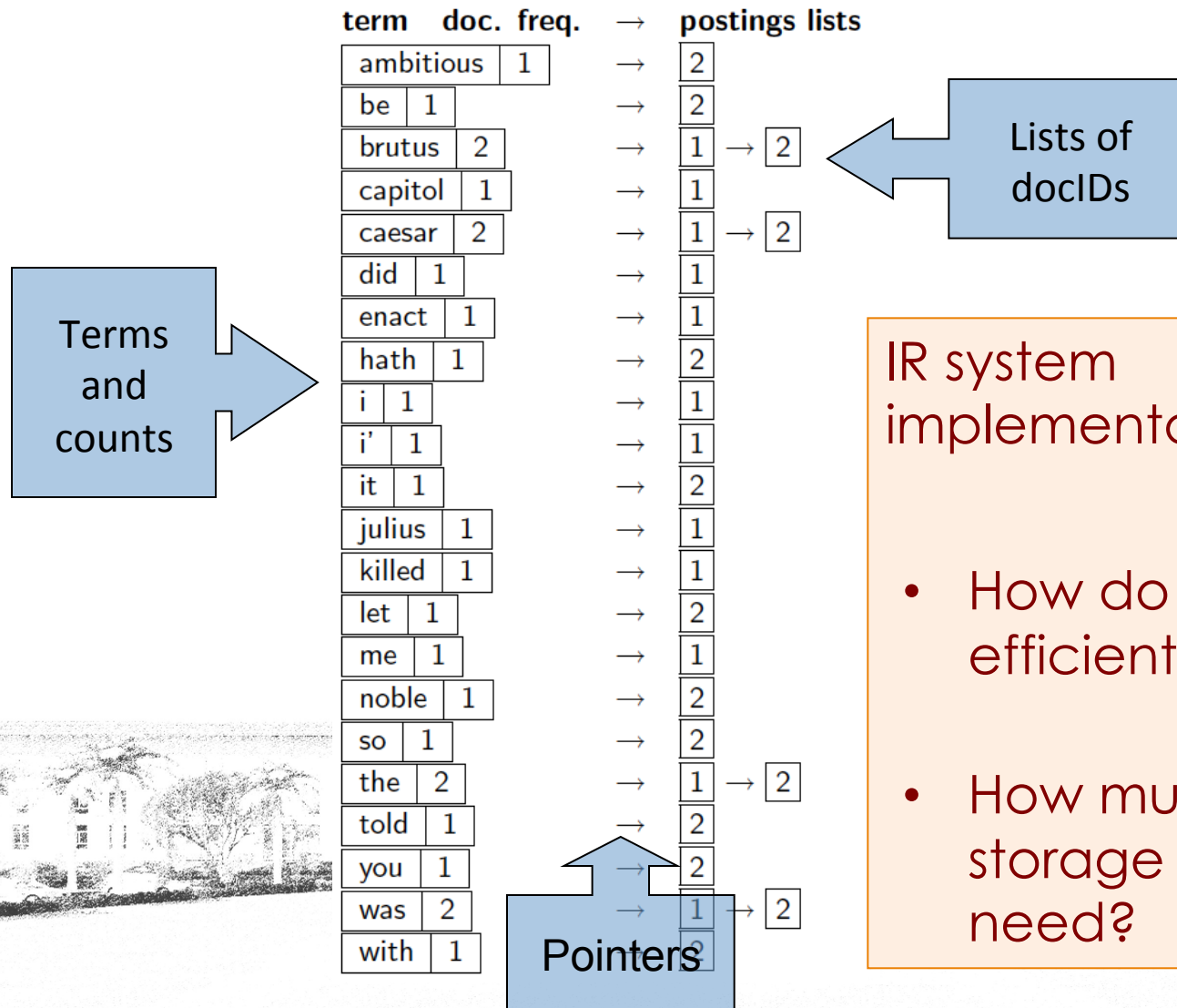
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2



# Where do we pay in storage?



IR system implementation

- How do we index efficiently?
- How much storage do we need?

# Exercise: Create Inverted Index

- d1) Turing machines can define computational processes that do not terminate. The informal definitions of algorithms generally require that the algorithm always terminates. This requirement renders the task of deciding whether a formal procedure is an algorithm impossible in the general case
- d2) Typically, when an algorithm is associated with processing information, data can be read from an input source, written to an output device and stored for further processing. Stored data are regarded as part of the internal state of the entity performing the algorithm.
- d3) For some such computational process, the algorithm must be rigorously defined: specified in the way it applies in all possible circumstances that could arise. Any conditional steps must be systematically dealt with, case-by-case

# Summary

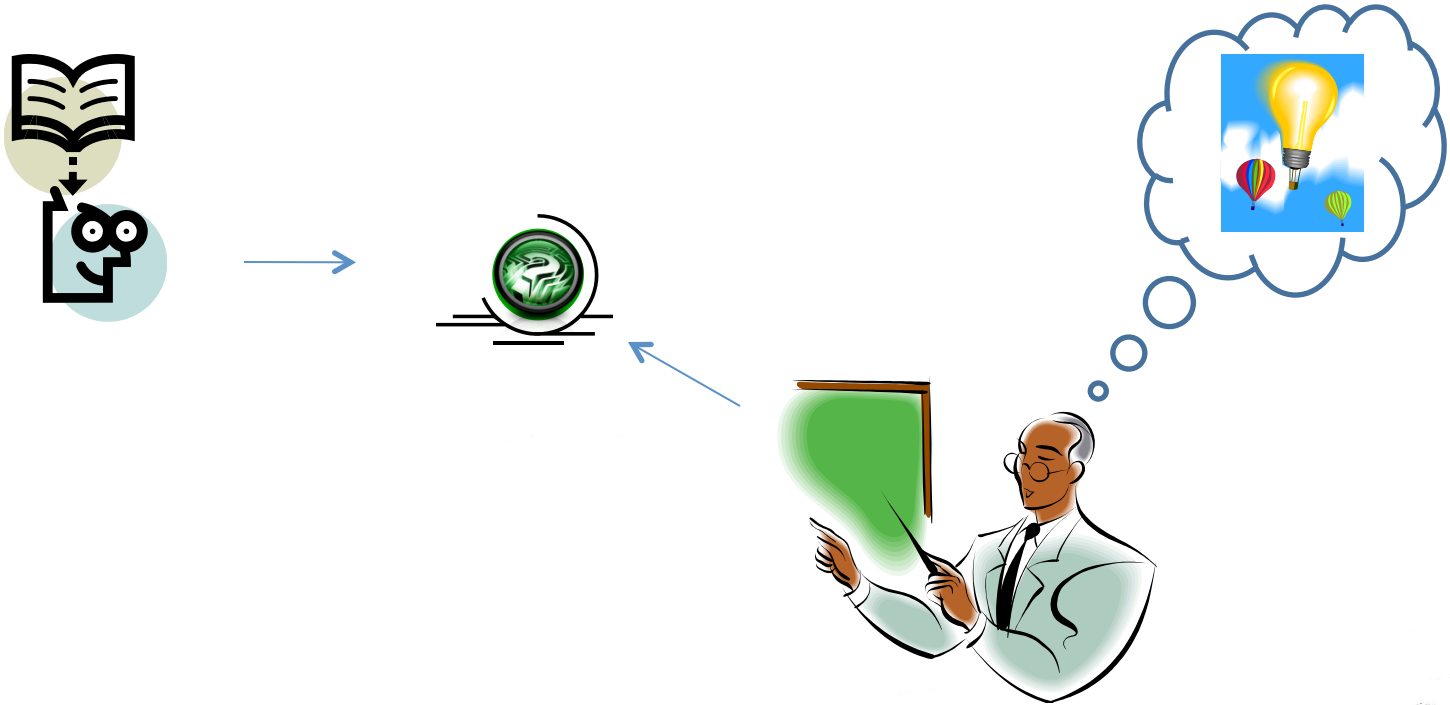
In this class, we focused on:

- (a) Words / Terms / Lexical Units
- (b) Tokenizing the terms
- (c) Preparing Term – Document matrix
- (d) Inverted Index Construction
  - i. Dictionary and Postings Lists
  - ii. Merging the Postings
  - iii. How much storage is required?

# 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 leave me an email any time  
(email is the best way to reach me faster)

# Thanks ...



## ... Questions ???