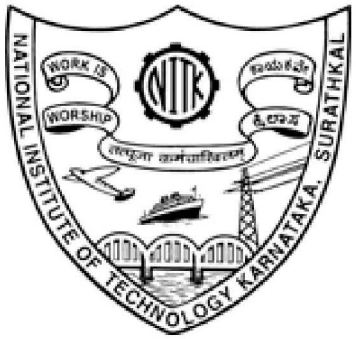
Jul – Nov 2022

IT458



Classic IR Models for Unstructured Text

Boolean Model, Inverted Index Construction

Boolean Retrieval

- A simple (and one of the oldest) IR model
  - based on set theory and Boolean algebra.

- Formalisms used -
  - Binary ranking function, i.e. 0/1-valued
  - Query language: Boolean expressions over index terms
  - Documents : subset of the vocabulary (index terms)

Boolean Retrieval

- Formalisms used -
  - *Documents* : a document is a subset of T (index terms)

$$T = \{t_1, t_2, \ldots, t_m\} \qquad D = \{D_1, \ldots, D_n\}$$

  - *Query language*: Boolean expressions over index terms

$$Q = (W_1 \vee W_2 \vee \cdots) \wedge \cdots \wedge (W_i \vee W_{i+1} \vee \cdots) \quad \text{when } t_i \in D_j$$

  - *Ranking*: Binary ranking function, i.e. 0/1-valued

Boolean Retrieval

- Representations used –
  - **Binary-term incidence matrix**
    - Considers only binary term-document frequencies.
  - **Inverted index**

Boolean Retrieval

Binary Term Incidence Matrix

- used as a way of indexing the corpus.
  - Efficient only for small corpora.

Documents →

Index Terms ↓

	HB1	HB2	HB3	HB4	HB5	HB6	HB7
Harry	1	1	1	1	1	1	1
Hermione	1	1	1	1	1	1	1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
Horcrux	0	0	0	0	0	1	1
Goblet	0	0	0	0	1	0	0
Riddikulus	0	1	0	0	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Boolean Retrieval

Binary Term Incidence Matrix

- Retrieval is based on membership in one or more sets using *Boolean Connectives* .
  - queries are written as *conjunctive* or *disjunctive* normal forms.

Boolean connectives

- Conjunction
- Disjunction
- Negation

∧	0	1
0	0	0
1	0	1

∨	0	1
0	0	1
1	1	1

¬	
0	1
1	0

Boolean Retrieval

Binary Term Incidence Matrix

Query 1 : "Harry AND Riddikulus AND NOT Horcrux"

= 1111111 AND 0100000 AND NOT (0000011)

= 1111111 AND 0100000 AND 1111100

= 0100000

Boolean Retrieval

Binary Term Incidence Matrix

Query 1 : "Harry AND Riddikulus AND NOT Horcrux"

= 1111111 AND 0100000 AND NOT (0000011)

= 1111111 AND 0100000 AND 1111100

= 0100000

Action: Retrieve documents matching the pattern

Boolean Retrieval - *Inverted Index*

- For larger corpora, documents are usually indexed by an *Inverted Index*.
  - For each term *t*, a list of all documents that contain *t* is stored.
  - enables fast query processing.

- Using the inverted index, queries of the type *"Show me all documents containing term X (AND/OR/NOT) Y)* "can be answered quickly.

Boolean Retrieval - Inverted Index Construction

- *Problem:* generate an inverted index for a document collection with some sample documents as below.
  - Doc 1 = "India's population continues to rise ….."
  - Doc 2= "that's one small step for a man, a giant leap for mankind…"
  - Doc 3 = "Healthcare providers up in strike …."

  …
  - Doc *n* = "Gandhi's small step was a turning point for India…"

Boolean Retrieval - Inverted Index Construction

- **Steps in Inverted Index construction –**
  - Tokenizing and preprocessing documents
    - Normalize list of document-wise tokens
  - Generate postings
  - Sort postings
  - Create postings lists, determine document frequency
  - Split the result into dictionary and postings file → INVERTED INDEX

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

1. Tokenization: Cut character sequence into word tokens.

- Example:

  Input: "India's population continues to rise ....."

  Output: Tokens

  India's

  population

  continues

  to

  rise....

- Each token is candidate for an index entry, after further processing.

Tokenization – some tricky cases

- India's → *India* AND *s*? *Indias*? *India's*?
- Hewlett-Packard → *Hewlett* and *Packard* as two tokens?
  - *state-of-the-art*: should we break up hyphenated sequence?
  - *co-education*
  - lowercase, lower-case, lower case ?
  - Will the user also put in possible hyphens while querying?
- San Francisco: one token or two?
  - How do you decide it is one token?

Tokenization – some tricky cases

- **Numbers**
  - *1/20/17 Jan. 20, 2017 20/1/17*
  - *55 B.C.*
  - *B-52*
  - *My PGP key is 324a3df234cb23e*

- *(800) 234-2333*
  - Often have embedded spaces
    - Older IR systems may not index numbers
    - Will often index "meta-data" separately
      - Creation date, format, etc.

2. Normalization:

- To bring all terms in indexed text as well as query to the same form.

- Most common normalization techniques -
  - Removal of periods in terms – e.g. ***U.S.A.*** and ***USA***
  - Case folding – e.g. ***USA*** and ***usa***.
  - Deleting hyphens to form a term – e.g. ***anti-terrorist*** and ***antiterrorist***
  - Spelling variants - e.g. ***favorite*** and ***favourite***
  - Accents – e.g. French ***résumé*** vs. ***resume.***

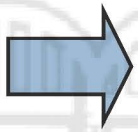
3. Stemming: Reduce terms to their "roots" before indexing.

- "Stemming" suggests crude affix chopping.
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

3. Stemming: Reduce terms to their "roots" before indexing.

- "Stemming" suggests crude affix chopping.
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.



for exampl compress and compress ar both accept as equival to compress

Other stemmers

EXTRAS

- Other stemmers :
  - Lovins stemmer
    - Single-pass, longest suffix removal (about 250 rules)
  - Paice/Husk stemmer
  - Snowball

Popular stemmers

EXTRAS

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

3. Lemmatization

- Reduce inflectional/variant forms to base form called a *lemma*.

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

3. Lemmatization

- Reduce inflectional/variant forms to base form called a *lemma*.
- closely related to stemming.
  - Difference -
    - Stemmer operates on a single word *without* knowledge of the context
    - Does not consider the different meanings of a word in usage.

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

3. Lemmatization

- E.g.,
  - The lemma of '*am*' → '*be*' *(similar to 'are', 'is' ...)*
  - Lemma of '*better*' → '*good*'
  - Lemma of '*walking*' → '*walk*' (i.e. stem & lemma are same)
  - Lemma of '*meeting*' → Context can be different (noun or verb)
    - e.g., "*in our last meeting*" or "*We are meeting again tomorrow*"

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

- **Stemming**

for example compressed and compression are both accepted as equivalent to compress. → *for exampl compress and compress ar both accept as equival to compress*

- **Lemmatization**

for example compressed and compression are both accepted as equivalent to compress. → ***for example compress and compression be both accept as equivalent to compress***

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

Stemming vs. Lemmatization

- Stemmers –
  - Typically easier to implement
  - Run faster.
  - For IR systems, stemming –
    - improves recall, or true positive rate.
    - reduces precision, or true negative rate.

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

Popular Lemmatizers –

- SpaCy
- WordNet Lemmatizer
- Python NLTK Lemmatizer
- LemmaGen

4. Stopword Removal:

- Using a stopword list, the most common words are often entirely eliminated from the vocabulary.
  - Words with very little semantic content: *the, a, and, to, be* .....

* Take a lot of space: ~30% of postings for top 30!

4. Stopword Removal:

- Often not preformed in many specialized applications -
  - Stopwords may be needed for cases like –
    - Phrase queries: "*King of Denmark*" , "*Government of India*"....

4. Stopword Removal:

- Often not preformed in many specialized applications -
  - Stopwords may be needed for cases like –
    - Phrase queries: "*King of Denmark*" , "*Government of India*"....
    - Various song titles, etc.: "*Let it snow*", "*Its my life*" ...

4. Stopword Removal:

- Often not preformed in many specialized applications -
  - Stopwords may be needed for cases like –
    - Phrase queries: "*King of Denmark*" , "*Government of India*"....
    - Various song titles, etc.: "*Let it snow*", "*Its my life*" ...
    - "Relational" queries: "*flights to London*" vs. "*flights from London*"

Boolean Retrieval - Inverted Index Construction

- *Problem:*
  - generate an inverted index for a document collection with some sample documents as below.
    - Doc 1 = "India's population continues to rise ….."
    - Doc 2= "that's one small step for a man, a giant leap for mankind…"
    - Doc 3 = "Healthcare providers up in strike …."

    …

    …

    - Doc *n* = "Gandhi's small step was a turning point for India…"

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

1. Tokenization

"India's", "population", "continues", "to", "rise"…..
"that's", "one", "small", "step", "for", "a", "man",
"a", "giant", "leap" …….
"Healthcare", " providers", "up", "in", "strike" ……
…
…

"Gandhi's", "small", "step". ….

Boolean Retrieval - Inverted Index Construction

Step 1: Initial stages of text processing

2. Normalization

"indias", "population", "continues", "to", "rise"…..
"thats", "one", "small", "step", "for", "a", "man",
"a", "giant", "leap" …….
"healthcare", " providers", "up", "in", "strike" ……
…
…

"gandhis", "small", "step". ….

3. Stemming:

```
indias population continues          india popul continu to
to rise thats one small              rise that one small step
step for a man a giant leap          for a man a giant leap
healthcare providers up in           healthcar provid up in
strike gandhis small step            strike gandhi small step
```

3. Lemmatizer:

```
indias population continues          india population continue
to rise thats one small              to rise that one small step
step for a man a giant leap          for a man a giant leap
healthcare providers up in           healthcare provider up in
strike gandhis small step            strike gandhis small step
```

4. Stopword Removal:

```
indias population continues
to rise thats one small
step for a man a giant leap
healthcare providers up in
strike gandhis small step
```

```
india popul continu rise
one small step man giant
leap healthcar provid
strike gandhi small step
```

Boolean Retrieval - Inverted Index Construction

Step 2: Generate term postings

- Step 2 : Generate term postings
  - Generate sequence of (Modified token, Document ID) pairs.

Boolean Retrieval - Inverted Index Construction

Step 2: Generate term postings

- Step 2 : Generate term postings
  - Generate sequence of (Modified token, Document ID) pairs.

Doc 1	Doc 2	
India's population continues to rise …..	that's one small step for a man, a giant leap for mankind…	…..

Boolean Retrieval - Inverted Index Construction

- Step 2 : Generate term postings
  - Generate sequence of (Modified token, Document ID) pairs.

Doc 1

India's population continues to rise …..

Doc 2

that's one small step for a man, a giant leap for mankind…

…..

Term	docID
india	1
popul	1
continue	1
to	1
rise	1
that	2
one	2
small	2
step	2
for	2
a	2
man	2
a	2
giant	2
leap	2
healthcar	3
provider	3
arm	3
.	.
.	.
.	.
.	.
.	.
.	.

Boolean Retrieval - Inverted Index Construction

- Step 3: Sort postings
  - Sort by terms (alphabetical)

also remove duplicates

Core indexing step

Term	docID
I	1
did	1
enact	1
gandhi	1
capital	1
I	1
was	1
killed	1
me	1
the	1
cancel	1
India	1
kill	1
me	1
so	2
let	2
it	2
be	2
with	2
India	2
the	2
noble	2
gandhi	2
haste	2
told	2
you	2
gandhi	2
was	2

Term	docID
ambiti	2
be	2
ball	1
blast	2
call	1
continue	1
capital	2
did	1
enact	1
gandhi	1
haste	1
i	2
it	1
india	2
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Boolean Retrieval - Inverted Index Construction

- Step 4: Create postings list and determine doc frequency
  - Multiple term entries in a single document are merged.

Term	docID
ambiti	2
be	2
ball	1
blast	2
call	1
continu	1
capital	2
did	1
enact	1
gandhi	1
haste	1
i	2
it	1
india	2
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.
ambiti	1
be	1
ball	2
blast	1
call	1
continue	1
capital	2
did	1
enact	1
gandhi	1
haste	1
i	1
it	1
india	1
killed	1
let	1
me	1
noble	1
so	1
the	2
told	1
you	1

Boolean Retrieval - Inverted Index Construction

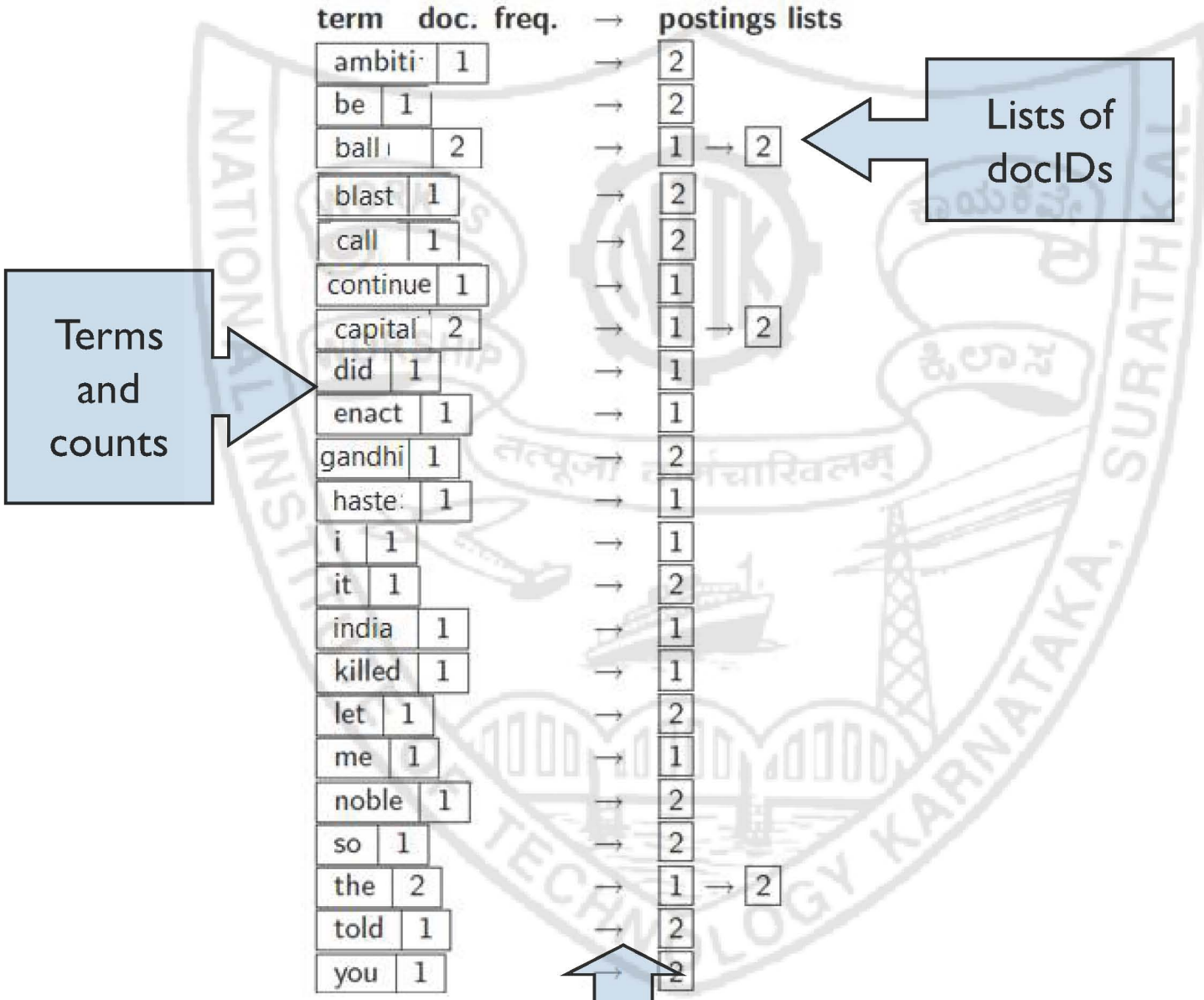
- Step 4: Create postings list and determine doc frequency
  - Multiple term entries in a single document are merged.

Term	docID
ambiti	2
be	2
ball	1
blast	2
call	1
continu	1
capital	2
did	1
enact	1
gandhi	1
haste	1
i	2
it	1
india	2
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
ambiti	1	→	2
be	1	→	2
ball	2	→	1 → 2
blast	1	→	2
call	1	→	2
continue	1	→	1
capital	2	→	1 → 2
did	1	→	1
enact	1	→	1
gandhi	1	→	2
haste	1	→	1
i	1	→	1
it	1	→	2
india	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

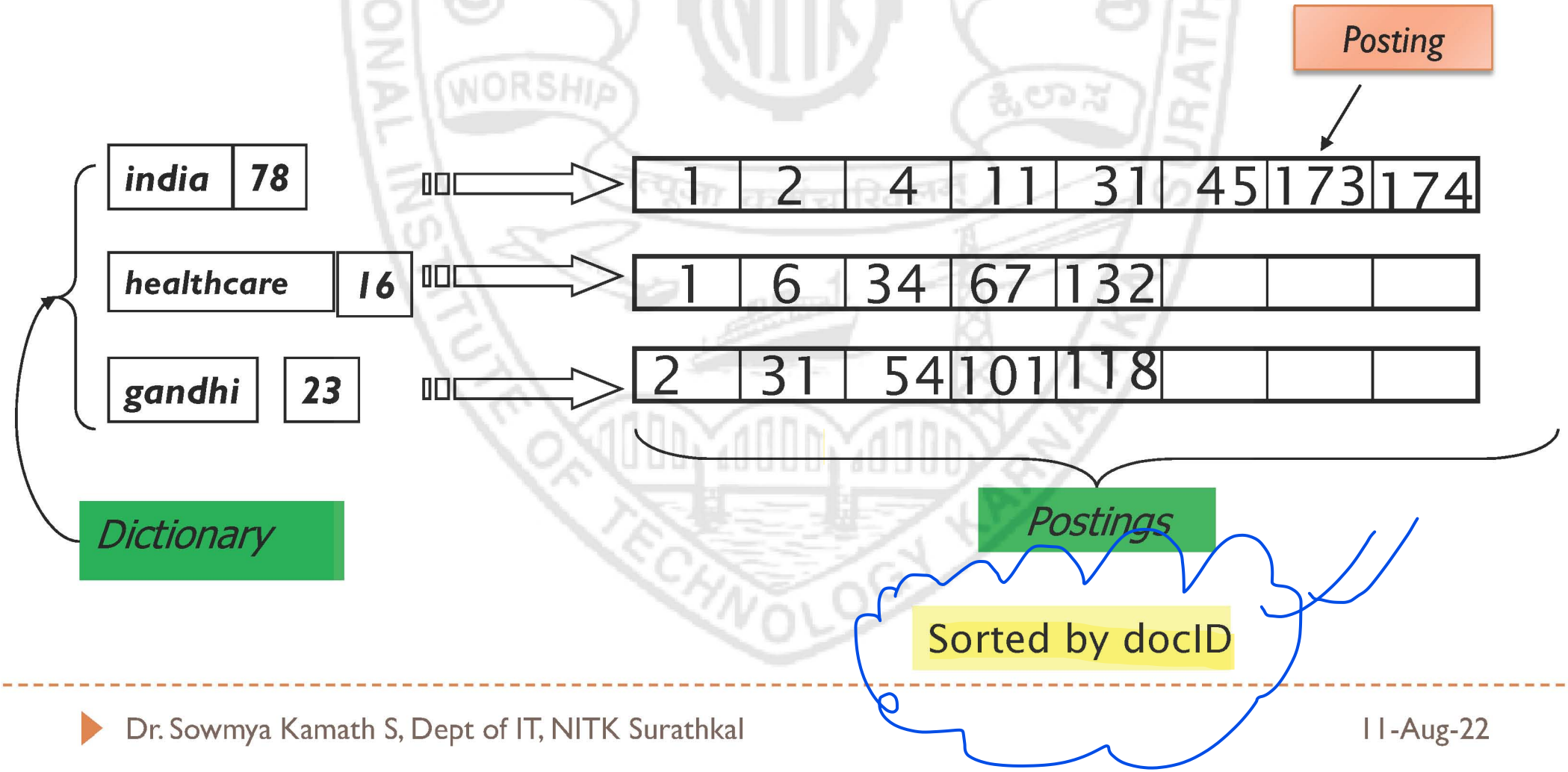
Boolean Retrieval - Inverted Index Construction

Step 4: Create postings list and determine doc frequency



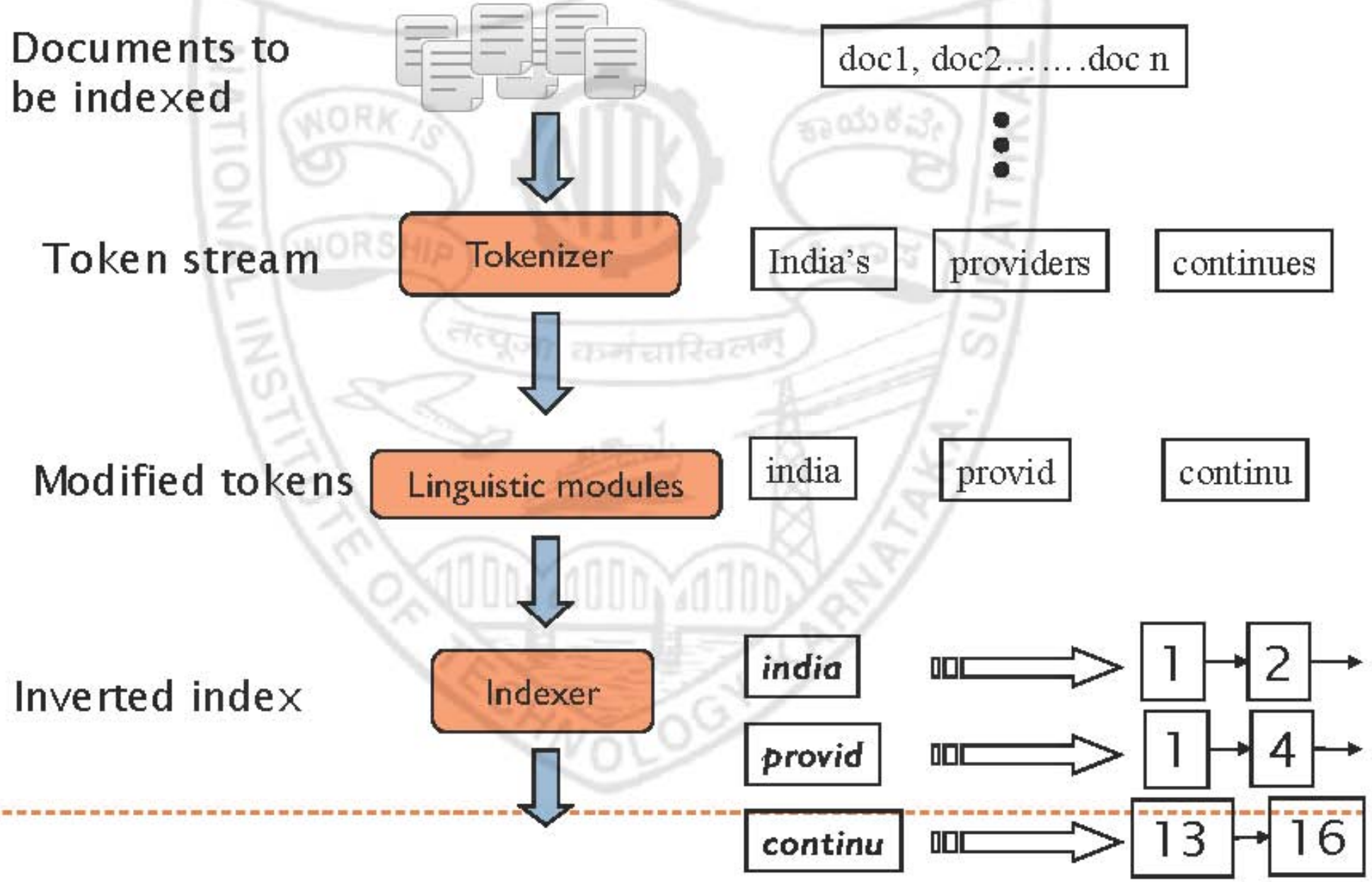
Boolean Retrieval - Inverted Index Construction

- Step 4: Create postings list and determine doc frequency
  - Finally, each term token is represented by variable-size postings lists



Boolean Retrieval

Inverted index construction process



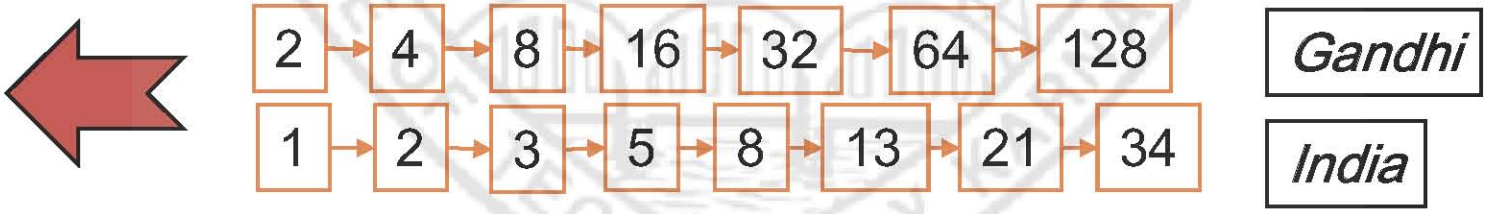
Boolean Retrieval - Inverted Index Construction

Boolean query processing

- Consider processing the query:

  Gandhi *AND* **India**

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

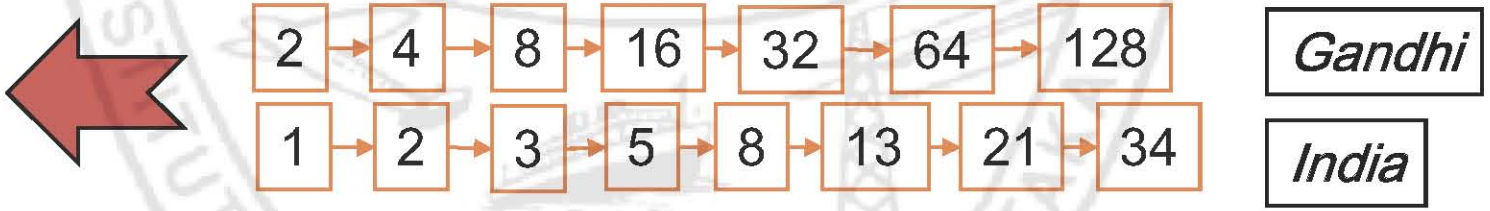


Boolean Retrieval - Inverted Index Construction

Boolean query processing

- **The merge operation**
  - If the list lengths are x and y, the merge takes $O(x+y)$ operations.

* Crucial: postings sorted by docID.



Boolean Retrieval - Inverted Index Construction

Boolean query processing

- For the sample document collection considered –
  - Doc 1 = "that's one small step for a man, a giant leap for mankind"
  - …
  - Doc *n* = "Gandhi's small step was a turning point for India"

- Query1 = "step AND mankind"
  - Result set: {Doc 1}

- Query2 = "step OR mankind"
  - Result set: {Doc 1, Doc *n*}

Boolean Retrieval - Inverted Index Construction

Boolean query processing

- To match natural language better, "BUT NOT" can be used instead of "AND NOT".
  - Query4 = "step BUT NOT india"

- Use "OF" to search for subsets of a given size:
  - Query5 = "2 OF {step, mankind, india}"

is equivalent to

  - Query5 = "(step AND mankind) OR (step AND india) OR (mankind AND india)"

IR Models - Boolean Retrieval (contd.)

- Pros:
  - Boolean expressions have precise semantics
  - Structured querying
  - For expert users, boolean queries can be very intuitive.
  - Simple and neat formalism → great attention in past years and was adopted by many of the early commercial bibliographic systems.

IR Models - Boolean Retrieval (contd.)

- Cons:
  - it is often not simple to translate an information need into a Boolean expression.
  - Most users find it difficult and awkward to express their query requests in terms of Boolean expressions.
  - No ranking.
  - No relevance feedback.

More reading...

- Radecki, T. (1983). Generalized Boolean methods of information retrieval. *International Journal of Man-Machine Studies*, *18*(5), 407-439.

- Salton, Gerard, Edward A. Fox, and Harry Wu. "Extended boolean information retrieval." *Communications of the ACM* 26.11 (1983): 1022-1036.