Basic IR Models

- Inverted Index
- Boolean querries
- Boolean and vector space retrieval model

More realistic scenario

Suppose corpus has 1 million documents(text)

Number of distinct term : 100,000

Now if we create Term Document Incidence matrix for this scenario

Number of cell in matrix $= 100{,}000 * 10{,}00{,}000$

$= 0.1 * 10^{12}$

$= 100GB$

(if one cell is stored in 1 byte of memory)

Term-Document Incidence Matrix is Sparse

	Process control block	Process scheduling	CPU utilization	Deadlock in operating system	Disk scheduling algorithm	Critical section
process	1	1	0	0	0	1
kernel	0	1	0	1	0	0
CPU	1	1	1	0	0	0
scheduling	0	1	0	0	1	0
deadlock	0	0	0	1	0	1

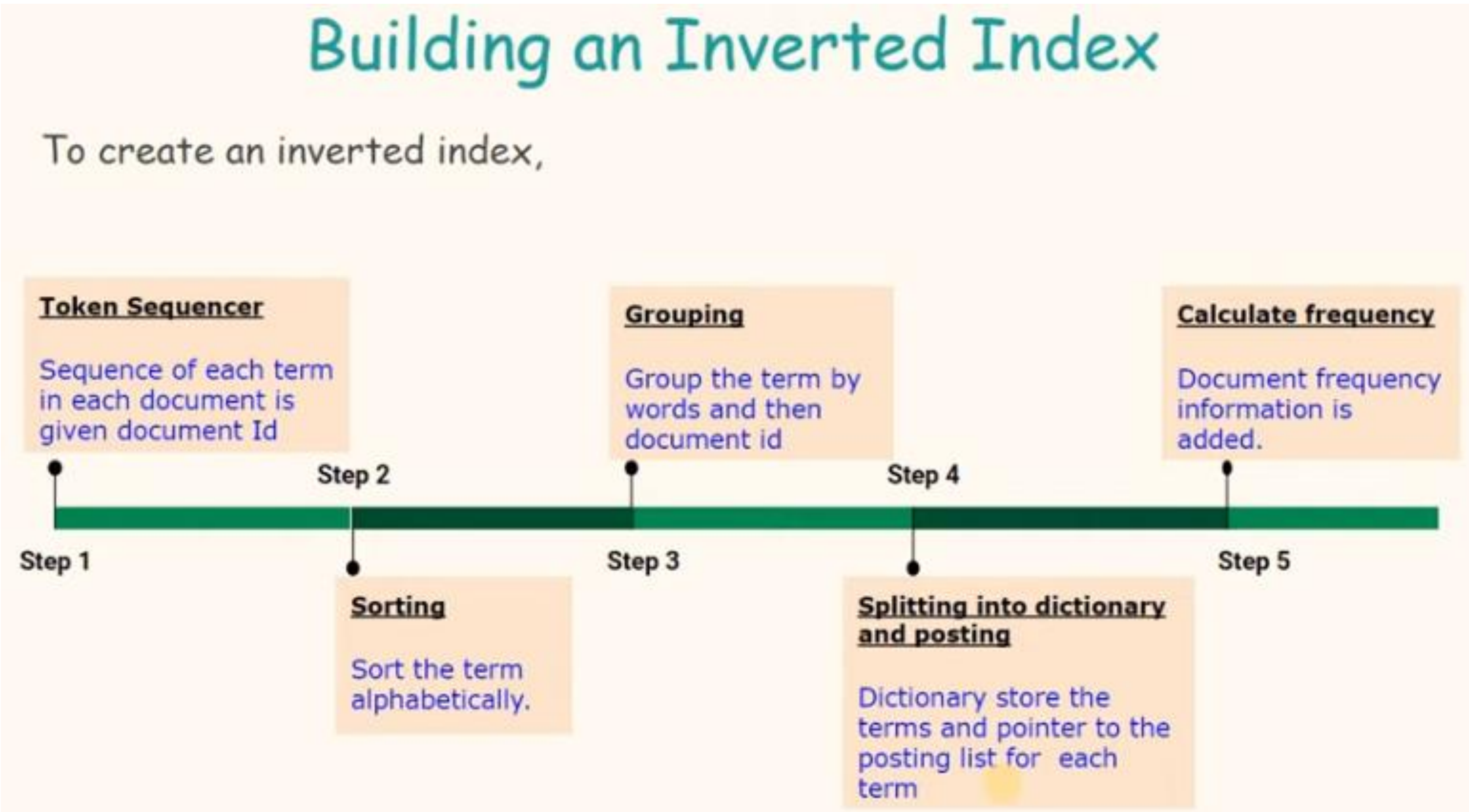
The matrix is converted into inverted index

What is an Inverted index ?

An inverted index is an index data structure storing a mapping from content, such as **words or numbers, to its locations in a document or a set of documents**.

It is a hashmap like data structure that directs you **from a word to a document or a web page**.

Inverted Index



Building an inverted index - Example

Doc1 :

Ram was average student in study but once he became friend of Shyam he started attending lecture regularly.

Doc2:

Shyam was enthusiastic student. He had helped his friend Ram a lot.Now Ram and Shyam are good friends.

Building an inverted index - Example

Term	DocID
Ram	1
was	1
average	1
student	1
in	1
study	1
but	1
once	1
he	1
became	1
friend	1
of	1
Shyam	1
he	1
started	1
attending	1
lecture	1
regularly	1

Term	DocID
Shyam	2
was	2
very	2
enthusiastic	2
student	2
He	2
had	2
helped	2
his	2
friend	2
Ram	2
a	2
lot	2
Now	2
Ram	2
and	2
Shyam	2
are	2
good	2
friend	2

Term	DocID
a	2
and	2
are	2
attending	1
average	1
became	1
but	1
enthusiastic	2
friend	1
friend	2
friend	2
good	2
had	2
he	1
he	1
He	2
helped	2
his	2
in	1
in	1
lecture	1

Term	DocID
lecture	1
lot	2
lot	2
Now	2
Now	2
of	1
once	1
Ram	1
Ram	2
Ram	2
regularly	1
Shyam	1
Shyam	2
Shyam	2
started	1
student	1
student	2
study	1
very	2
was	1
was	2

Building an inverted index - Example

Term	DocID
a	2
and	2
are	2
attending	1
average	1
became	1
but	1
enthusiastic	2
friend	1
friend	2
friend	2
good	2
had	2
he	1
he	1
He	2
helped	2
his	2
in	1
lecture	1
lot	2

Term	DocID
Now	2
in	1
lecture	1
lot	2
Now	2
of	1
once	1
Ram	1
Ram	2
Ram	2
regularly	1
Shyam	1
Shyam	2
Shyam	2
started	1
student	1
student	2
study	1
very	2
was	1
was	2

Term	Doc. Frequency	Posting List
a	1	→ 2
and	1	→ 2
are	1	→ 2
attending	1	→ 1
average	1	→ 1
became	1	→ 1
but	1	→ 1
enthusiastic	1	→ 2
friend	2	→ 1 → 2
good	1	→ 2
had	1	→ 1
he	1	→ 1
He	1	→ 2
helped	1	→ 2
his	1	→ 2
in	1	→ 1
lecture	1	→ 1
lot	1	→ 2
Now	1	→ 2
of	1	→ 1
once	1	→ 1
Ram	2	→ 1 → 2
regularly	1	→ 1
Shyam	2	→ 1
started	1	→ 1
student	2	→ 1
very	1	→ 2
was	2	→ 1

Boolean Retrieval Query

Boolean retrieval model : In this model we represent query which is in boolean expressions of terms.

Terms are combined with AND , NOR , NOT

X AND Y: represents doc that contains both X and Y.

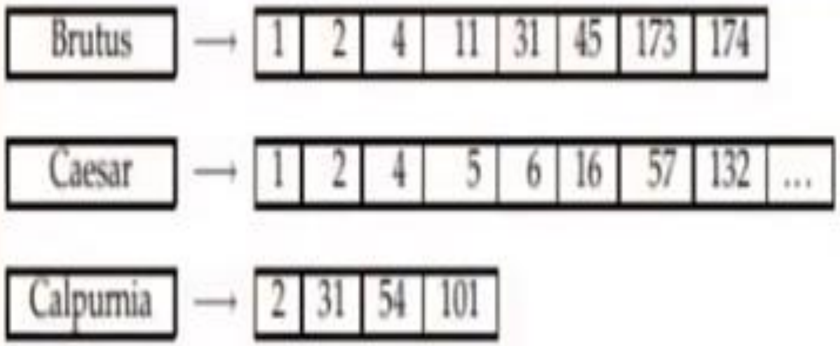
X OR Y: represents doc that contains either X or Y.

Processing Boolean queries

Brutus AND Calpurnia

Over the invertwd index we:

1. Locate Brutus in the Dictionary
2. Retrieve its postings
3. Locate Calpurnia in the Dictionary
4. Retrieve its postings
5. Intersect the two postings lists, as shown below:



Implementation(Brutus AND Calpurnia)

```
INTERSECT(p1, p2)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
      then ADD(answer, docID(p1))
           p1 ← next(p1)
           p2 ← next(p2)
      else if docID(p1) < docID(p2)
             then p1 ← next(p1)
             else p2 ← next(p2)
return answer
```

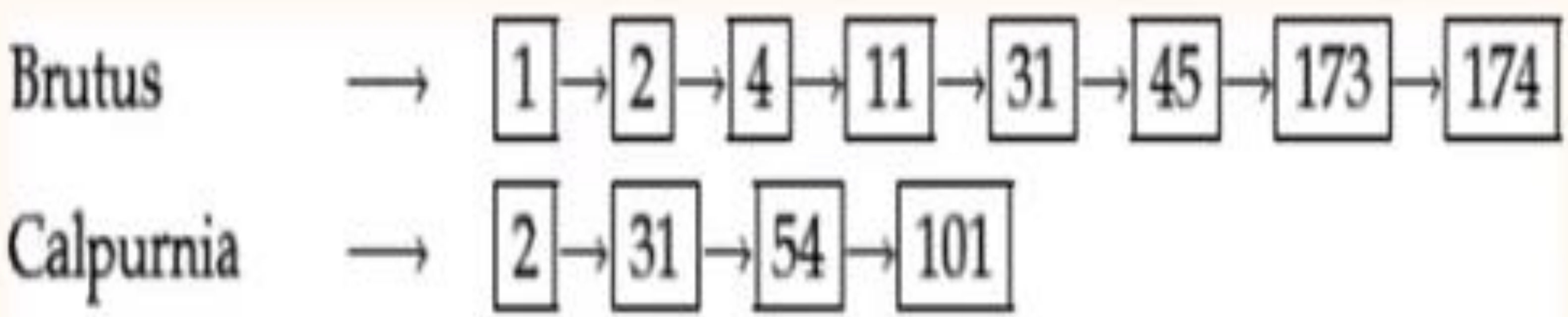
Brutus → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

Calpurnia → 2 → 31 → 54 → 101

Answer based on intersection

2	**31**

Implementation(Brutus OR Calpurnia)



Answer based on Union

1	2	4	11	31	45	54	101	173	174

Implementation(NOT Brutus)

Brutus $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4}$

Answer : based on NOT

Vector Space Model

- The vector space model is a way of representing documents through the words they contain
- It is standard technique in information retrieval
- The VSM allows decisions to be made about which documents are similar to each other and to keyword queries.

Vector Space Model

- In this model, documents and queries are assumed to be part of a t-dimensional vector space, where t is the number of index terms (words, stems, phrases, etc.).
- A document Di is represented by a vector of index terms:
- Di = (di1, di2, . . . , dit),
- where dij represents the weight of the jth term.

- A document collection containing n documents can be represented as a matrix of term weights, where each row represents a document and each column describes weights that were assigned to a term for a particular document:

	$Term_1$	$Term_2$	. . .	$Term_t$
Doc_1	d_{11}	d_{12}	. . .	d_{1t}
Doc_2	d_{21}	d_{22}	. . .	d_{2t}
.	.			
Doc_n	d_n 1	d_{n2}	. . .	d_{nt}

- The **term-document matrix** has been rotated so that now the terms are the rows and the documents are the columns. The term weights are simply the count of the terms in the document.

D_1 Tropical Freshwater Aquarium Fish.
D_2 Tropical Fish, Aquarium Care, Tank Setup.
D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	Documents			
	D_1	D_2	D_3	D_4
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

Document D3, for example, is represented by the vector (1, 1, 0, 2, 0, 1, 0, 1, 0, 0, 1).

- **Queries** are represented the same way as documents. That is, a query Q is represented by a vector of t weights:
- Q = (q1, q2, . . . , qt),
- where qj is the weight of the jth term in the query.
- If, for example the query was “tropical fish”, then using the vector representation in, the query would be (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1).

D_1 Tropical Freshwater Aquarium Fish.
D_2 Tropical Fish, Aquarium Care, Tank Setup.
D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

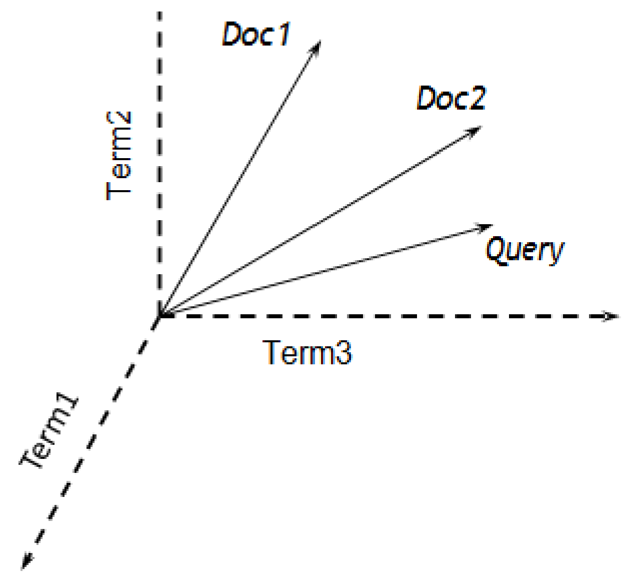
Terms	Documents			
	D_1	D_2	D_3	D_4
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

If, for example the query was “tropical fish”, then using the vector representation in, the query would be (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1).

- A **similarity measure is** used (rather than a distance or dissimilarity mea- sure), so that the documents with the highest scores are the most similar to the query. The most successful of these is the cosine correlation similarity measure.
- The cosine correlation measures the cosine of the angle between the query and the document vectors.

- Given this representation documents can be ranked by computing the distance between the points representing the documents and query.

▪ The documnts with the highest score are the most Similar to query



Vector Space retrieval Model

- In this model documents and queries are assumed to be part of t dimensional vector space, where t is the no. of index terms (words,phrase etc)
- A document D_i is represented by a vector of index terms:

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} d_{ij}^2 \cdot \sum_{j=1}^{t} q_j^2}}$$

- As an example, consider two documents
- D1 = (0.5, 0.8, 0.3) and
- D2 = (0.9, 0.4, 0.2) indexed by three terms, where the numbers represent term weights. Given the query Q = (1.5, 1.0, 0) indexed by the same terms, the cosine measures for the two documents are:

- Cosine(D1,Q)=0.87
- Cosine(D2,Q)=0.97
- Since the score is high for document D2
- So we can say document D2 is more relavant to query Q.

TF-IDF(Term frequency-inverse document frequency)

What is TF-IDF(term frequency-inverse document frequency)?

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

- → The term frequency of a word in a document(t_f)
- → The inverse document frequency(idf) of the word across a set of documents.The closer it is to 0, the more common a word is.

How TF-IDF calculated

- Tf*idf results in TF-IDF score of a word in a document.The higher the score,the more relavant that word in that particular document.
- Mathemetical notation:

$$tf\,idf\,(t, d, D) = tf\,(t, d)\ .\ idf\,(t, D)$$

$$tf\,idf\,(t, d, D) = tf\,(t, d)\;.\;idf\,(t, D)$$

t=term
d=document
D-set of documents
tf=term frequency

- tf-idf is a weighting scheme that assigns each term in a document a weight based on its term frequency (tf) and inverse document frequency (idf). The terms with higher weight scores are considered to be more important.

- Modern information retrieval by Ricardo Baeza edition 2 page no.100

Vector Space retrieval Model

Inverted index

Tf-ifd

Stemming