Savitribai Phule Pune University
Final Year of Information Technology (2019 Course)
(With effect from Academic Year 2022-23)

Semester VII

Course Code	Course Name	Teaching Scheme(Hours/week)			Examination Scheme and Marks						Credit Scheme			
		Lecture	Practical	Tutorial	Mid-Sem	End-Sem	Termwork	Practical	Oral	Total	Lecture	Practical	Tutorial	Total
414441	Information and Storage Retrieval	03	-	-	30	70	-	-	-	100	3	-	-	3
414442	Software Project Management	03	-	-	30	70	-	-	-	100	3	-	-	3
414443	Deep Learning	03	-	-	30	70	-	-	-	100	3	-	-	3
414444	Elective III	03	-	-	30	70	-	-	-	100	3	-	-	3
414445	Elective IV	03	-	-	30	70	-	-	-	100	3	-	-	3
414446	Lab Practice III	-	04	-	-	-	25	-	25	50	-	2	-	2
414447	Lab Practice IV	-	02	-	-	-	25	25	-	50	-	1	-	1
414448	Project Stage-I	-	-	02	-	-	50	-	-	50	-	-	2	2
414449	Audit Course7													
										Total Credit	15	03	02	20
	Total	15	06	02	150	350	100	25	25	650	15	03	02	20

Elective III:

- Mobile Computing
- High Performance Computing
- Multimedia Technology
- Smart Computing

Elective IV:

- Bioinformatics
- Introduction to DevOps
- Computer Vision
- Wireless Communications

Lab Practice-III:

It is based on subjects:

- Information and Storage Retrieval

Lab Practice-IV:

It is based on subjects:

- Deep Learning

B.E. (INFORMATION TECHNOLOGY)

SEMESTER-VII

Time– 2.00 p.m. to 3.00 p.m.

Day& Date	SUBJECT (2019 Course)	SUBJECT CODE
Tuesday 19/08/2025	Information and Storage Retrieval	414441
Wednesday 20/08/2025	Software Project Management	414442
Thursday 21/08/2025	Deep Learning	414443
Friday 22/08/2025	(ELECTIVE-III) Mobile Computing	414444A
	(ELECTIVE-III) High Performance Computing	414444B
	(ELECTIVE-III) Multimedia Technology	414444C
	(ELECTIVE-III) Smart Computing	414444D
Saturday 23/08/2025	(ELECTIVE-IV)Bioinformatics	414445A
	(ELECTIVE-IV) Introduction to DevOps	414445B
	(ELECTIVE-IV) Computer Vision	414445C
	(ELECTIVE-IV) Wireless Communications	414445D

COURSE CONTENTS		
Unit I	**Introduction to Information Retrieval**	**(06 hrs)**
Basic Concepts of IR, Data Retrieval & Information Retrieval, Text mining and IR relation, IR system block diagram, **Automatic Text Analysis:** Luhn's ideas, Conflation Algorithm, Indexing and Index Term Weighting, Probabilistic Indexing, Automatic Classification. Measures of Association, Different Matching Coefficients, Cluster Hypothesis, **Clustering Techniques: Rocchio's** Algorithm, Single pass algorithm, Single Link algorithm.		
Mapping of Course Outcomes for Unit I	**CO1**	
Unit II	**Indexing and Searching Techniques**	**(06 hrs)**
Indexing: Inverted file, Suffix trees & suffix arrays, Signature Files, Scatter storage or hash addressing. **Searching Techniques:** Boolean Search, sequential search, Serial search, cluster-based retrieval, Query languages, Types of queries, Patterns matching, structural queries. **IR Models:** Basic concepts, Boolean Model, Vector Model, Probabilistic Model.		
Mapping of Course Outcomes for Unit II	**CO2**	

Q3) a) Explain the concepts of signature files in information retrieval. **[5]**

b) Explain exhaustivity & specificity with respect to index term weighting. **[6]**

c) List and explain the types of queries. **[4]**

OR

Q4) a) Explain the concept of inverted index file. How it can be used in information retrieval. **[6]**

b) Explain the different kinds of search strategies. **[6]**

c) Discuss vector model for information retrieval. **[3]**

Q3) a) Demonstrate the application of the Boolean search technique using a set of documents and a complex Boolean query involving multiple operators (AND, OR, NOT). **[5]**

b) Explain the concept of Inverted index file. How it can be used in Information Retrieval. **[6]**

c) List and explain the types of queries. **[4]**

OR

Q4) a) Explain exhaustivity and specificity with respect to Index term weighting. **[4]**

b) Compare and contrast the basic concepts, strengths, and limitations of the Boolean Model, Vector Model, and Probabilistic Model. Provide insights into when and why each model would be chosen to optimize search results, considering factors like ranking accuracy and complexity. **[8]**

c) Elaborate cluster-based retrieval in brief. **[3]**

Q3) a) Explain the concept of suffix trees in information retrieval. **[5]**

b) Explain the different kinds of searching techniques in IR. **[6]**

c) Write a short note on probabilistic model. **[4]**

OR

Q4) a) Explain concept of inverted index file. How can it be used in information retrieval. **[6]**

b) Explain various IR model in detail with their advantages & disadvantages. **[9]**

1. Indexing:

- Inverted file
- Suffix trees & suffix arrays
- Signature Files
- Scatter storage or hash addressing

2. Searching Techniques:

- Boolean Search
- Sequential search
- Serial search
- Cluster-based retrieval
- Query languages
- Types of queries
- Pattern matching
- Structural queries

3. IR Models:

- Basic concepts
- Boolean Model
- Vector Model
- Probabilistic Model

1. Indexing:
- Inverted file
- Suffix trees & suffix arrays
- Signature Files
- Scatter storage or hash addressing

a) Explain the concepts of signature files in information retrieval. **[5]**

Explain the concept of Inverted index file. How it can be used in Information Retrieval. **[6]**

a) Explain the concept of suffix trees in information retrieval. **[5]**

1. Indexing

Indexing ka matlab hota hai data ko aise organize karna ki search fast ho jaye — jaise kitaab ka “Index page” jahan se page number se direct content mil jata hai.

Inverted File

Inverted Index ek aisa data structure hai jo word → document list ka mapping store karta hai.
Iska naam *inverted* isliye hai kyunki normally hum documents ke andar words store karte hain,
par yaha hum words ke saath un documents ka list store karte hain jisme wo word aata hai.

Structure
- **Term (Word) → Posting List** (kaunse documents me hai aur kis position pe).

Example
Maan lo humare paas 3 documents hain:

Document	Content
Doc1	Apple is red
Doc2	Banana is yellow
Doc3	Apple and banana are fruits

Inverted Index Table:

Term	Posting List
Apple	Doc1, Doc3
Banana	Doc2, Doc3
Red	Doc1
Yellow	Doc2
Fruits	Doc3

Use in Information Retrieval (IR)

•Jab user search kare "Apple", system directly inverted index table me Apple term dhundta hai aur uska posting list (Doc1, Doc3) de deta hai.
•Isse har document me search karne ki zarurat nahi padti, aur search fast ho jata hai.
•Google, Bing jaise search engines isi method ka use karte hain.

Advantages
1.Search bahut **fast** ho jata hai.
2.Storage me kam jagah lagti hai.
3.Boolean search (AND, OR, NOT) easily implement hota hai.

1. Suffix Tree

Concept

- **Suffix Tree ek compressed trie hota hai jo kisi string ke saare suffixes store karta hai.**
- **Iska use mainly substring searching, pattern matching, aur string operations ko fast karne me hota hai.**
- **Compressed trie ka matlab hai ki continuous same path ko ek hi edge me store karte hain (memory efficient).**

Example

String: `"banana$"` (`$` end symbol use hota hai taaki unique suffix ban sake)

Suffixes:

1. banana$
2. anana$
3. nana$
4. ana$
5. na$
6. a$
7. $

Structure Idea (simplified):

Root se har suffix ka path hota hai, common prefix wale paths merge hote hain.
Kaam ka fayda: Agar tumhe "ana" search karna hai, to tum root se "a" → "na" ka path follow karoge aur direct match mil jayega without checking har position.

Use
•Fast **substring search** (O(m) time, where m = pattern length)
•DNA sequence matching, text editors me “find” function

2. Suffix Array

Concept

- Suffix Array ek **sorted array** hota hai jisme string ke saare suffixes ke **starting positions** store hote hain.
- Ye Suffix Tree ka space-efficient alternative hai.

Example

String: `"banana"`

Suffixes with indexes:

0: banana

1: anana

2: nana

3: ana

4: na

5: a

Sorted Suffixes:

- 5: a
- 3: ana
- 1: anana
- 0: banana
- 4: na
- 2: nana

Suffix Array: `[5, 3, 1, 0, 4, 2]`

Feature	Suffix Tree	Suffix Array
Structure	Compressed Trie	Sorted array of suffix indexes
Speed	Faster (O(m))	Slower (O(m log n))
Space	More space	Less space
Use	Direct pattern search	Space-efficient pattern search

•Signature Files

Signature Files in Information Retrieval:

Concept

•Signature Files ek **filtering method** hai jo Information Retrieval me use hoti hai.

•Har document ka ek **compact representation (signature)** banate hain — usually bit string form me.

•Query search karte waqt pehle **signature compare** hota hai, fir agar match mila to actual document check hota hai.

•Isse unnecessary document scanning kam hota hai → retrieval fast hota hai.

How it Works

1.Har document ke important words ka **hash code** banta hai.
2.Hash code bits ko combine karke ek **fixed-size bit string** banate hain → isse signature kehte hain.
3.Jab user query deta hai, us query ka bhi signature banta hai.
4.Signature compare hota hai documents ke signatures ke saath.
5.Jo match hote hain unhi documents me actual detailed search hota hai.

Example

Maan lo do documents hain:

Doc1: "Apple is red"

Doc2: "Banana is yellow"

- Apple ka hash → 101000
- Banana ka hash → 010100
- "Apple is red" ka signature → 101011
- "Banana is yellow" ka signature → 010111

Query: `"Apple"` → signature = 101000

Signature compare karke Doc1 mil jayega → usme actual check hoga.

Advantages

- Document search fast ho jata hai.
- Storage efficient hota hai.
- Large database me scanning kam hoti hai.

•Scatter storage or hash addressing

Scatter Storage or Hash Addressing

Concept

- **Hash Addressing** ek method hai jisme hum ek **hash function** use karke data ka **direct storage location** nikalte hain.
- Isme data randomly (scatter) different storage locations me save hota hai, but **hash function** se hume pata chal jata hai ki data kahan rakha hai.
- Ye **fast access** ke liye use hota hai — direct jump to data location.

How it Works

1. **Key** (unique identifier) lete hain.
2. Us key par **hash function** lagate hain → ye ek number (address) return karta hai.
3. Data us address/location par store hota hai.
4. Jab search karte hain, wahi hash function lagake direct address milta hai.

Example

Suppose hum student roll numbers store kar rahe hain:

- **Hash function:** `address = roll_no % 10`

Roll No	Name	Address (Hash)
101	Amit	1
105	Ravi	5
112	Sita	2

Search: Roll No = 105 → `105 % 10 = 5` → directly address 5 par data mil gaya.

Collision Handling

Kabhi-kabhi do keys same address de dete hain → ise **collision** kehte hain.

Collision handle karne ke methods:

- **Chaining** (same address pe linked list banani)
- **Open Addressing** (next free slot find karna)

Advantages

- Very **fast search** (O(1) average case)
- Easy to implement
- Useful in large databases for quick retrieval

2. Searching Techniques:

- Boolean Search
- Sequential search
- Serial search
- Cluster-based retrieval
- Query languages
- Types of queries
- Pattern matching
- Structural queries

Q3) a) Demonstrate the application of the Boolean search technique using a set of documents and a complex Boolean query involving multiple operators (AND, OR, NOT). **[5]**

c) Elaborate cluster-based retrieval in brief. **[3]**

c) List and explain the types of queries. **[4]**

b) Explain the different kinds of search strategies. **[6]**

b) Explain the different kinds of searching techniques in IR. **[6]**

Boolean Search – Concept :

•Boolean Search ek searching technique hai jisme **logical operators** ka use hota hai to filter documents.
•Ye operators **Boolean logic** pe based hote hain:

- **AND** → dono terms hone chahiye document me
- **OR** → koi ek term hone chahiye
- **NOT** → ek term hona chahiye, doosra nahi hona chahiye

Operator	Meaning	Example Query	Result
AND	Dono keywords hone chahiye	apple AND mango	Sirf wo docs jisme dono hain
OR	Koi ek keyword chalega	apple OR mango	Docs jisme apple ya mango ya dono hain
NOT	Ek keyword hoga, doosra nahi	apple NOT mango	Apple wale docs, lekin mango wala nahi

Documents:

- **D1:** "Apple and Mango are fruits"
- **D2:** "Apple is red"
- **D3:** "Mango is yellow"
- **D4:** "I like Apple and Banana"

Query: `(Apple AND Mango) OR (Apple AND NOT Banana)`

Step-by-step:

1. **Apple AND Mango** → D1 only (both words present)
2. **Apple AND NOT Banana** → D2 (apple yes, banana no) and D1 (banana no)
3. Combine with OR:
    - Result = D1 (from step 1) + D2 (from step 2)
    - **Final Output: D1, D2**

Doc	Contains Apple	Contains Mango	Contains Banana	Matches Query?
D1	Yes	Yes	No	✅
D2	Yes	No	No	✅
D3	No	Yes	No	❌
D4	Yes	No	Yes	❌

•Sequential search

Sequential Search – Concept

- Isko **Linear Search** bhi bolte hain.
- Data ko ek ek karke **sequence me** check kiya jata hai jab tak required item mil na jaye ya list khatam ho jaye.
- Ye method **unsorted ya sorted** dono tarah ke data me kaam karta hai.
- **Drawback:** Large dataset me slow hota hai, kyunki har element check karna padta hai.

Steps:

1. Start from **first item**.
2. Compare with search key.
3. Agar match milta hai → stop, item found.
4. Agar nahi milta → next item par jao.
5. Agar end tak nahi mila → item not found.

Example:

Data List: [12, 45, 78, 34, 90]

Search key: 34

- Step 1: Compare 12 with 34 → no match
- Step 2: Compare 45 with 34 → no match
- Step 3: Compare 78 with 34 → no match
- Step 4: Compare 34 with 34 → match found ✅

Advantages:

- Simple to implement
- Works on unsorted data

Disadvantages:

- Time-consuming for large datasets (O(n) complexity)

•Serial search

Serial Search – Concept

- Isme bhi elements ko **ek ke baad ek** check kiya jata hai (serially).
- Data ko **starting se end tak** compare karte hain jab tak required value mil na jaye ya data khatam ho jaye.
- Use **sorted** ya **unsorted** dono data sets me kiya ja sakta hai.

Steps:

1. First element se shuru karo.
2. Compare karo search value se.
3. Agar match milta hai → Stop, item found.
4. Agar match nahi mila → next element par jao.
5. Agar list end ho gayi aur match nahi mila → item not found.

Example:

Data: [8, 15, 27, 32, 40]

Search key: 27

- Step 1: Compare 8 with 27 → no match
- Step 2: Compare 15 with 27 → no match
- Step 3: Compare 27 with 27 → match found ✅

Note:

"Serial Search" and "Sequential Search" are **same technique** — bas naam ka difference hai.

c) Elaborate cluster-based retrieval in brief. [3]

Cluster-Based Retrieval

Cluster-based retrieval ek *information retrieval* technique hai jisme documents ko similar content ya topics ke basis par groups (clusters) me arrange kiya jata hai. Idea ye hai ki *similar documents ek saath rakhe jayein*, taki search process efficient aur relevant ho.

Process

1. **Document Representation** – Har document ko ek vector form me represent kiya jata hai (jaise term frequency, TF-IDF values).
2. **Clustering** – Similarity measures (cosine similarity, Euclidean distance) ka use karke documents ko clusters me group kiya jata hai. Common algorithms:
   - *K-means clustering*
   - *Hierarchical clustering*
3. **Query Processing** – Jab user query deta hai:
   - Pehle query ka relevant cluster find hota hai.
   - Fir us cluster ke documents retrieve hote hain.

Advantages

- Search space chhota ho jata hai → faster retrieval.
- Zyada relevant results milte hain.
- Useful jab dataset bada aur diverse ho.

Example

Document collection:

- D1: “India won cricket match”
- D2: “Virat Kohli scores century”
- D3: “Messi wins football award”
- D4: “FIFA World Cup results”

Clusters:

- **Cluster 1 (Cricket)** → D1, D2
- **Cluster 2 (Football)** → D3, D4

Query: “cricket century” → system sirf *Cluster 1* search karega → fast aur relevant output.

•Query languages

Query Languages in Information Retrieval

Definition
Query language ek formal language hai jo users ko *database* ya *information retrieval system* me search queries likhne ki facility deta hai.
Ye specify karta hai ki kya information chahiye aur kaise retrieve karni hai.

Types of Query Languages

1. **Boolean Query Language**
   - Based on **Boolean logic** (AND, OR, NOT).
   - Example: `"cricket AND century"` → dono words wala result aayega.
2. **Structured Query Language (SQL)**
   - Mainly relational databases ke liye.
   - Example: `SELECT * FROM students WHERE marks > 80;`
3. **Natural Language Queries**
   - Human language me query likhna.
   - Example: `"Show me all cricket matches won by India"`
   - IR system ko interpret karna padta hai.
4. **Vector Space Query Language**
   - Queries ko **vector form** me represent karke similarity measure use hota hai.
   - Example: cosine similarity between query and document vectors.

Key Features

- User ko precise search karne ka option deta hai.
- Support for filters, ranking, and advanced search operators.
- Can be formal (SQL, Boolean) ya informal (natural language).

Example

Suppose hume cricket aur Virat Kohli related documents chahiye, lekin football wale nahi:

- Boolean query: ("cricket" AND "Virat Kohli") NOT "football"
- SQL query:

```
SELECT * FROM sports
WHERE game = 'cricket' AND player = 'Virat Kohli';
```

Pattern Matching

Definition

Pattern matching ek process hai jisme hum ek **pattern** (word, phrase, symbol sequence) ko **data/document** me search karte hain aur uske positions ya matches detect karte hain.

Types of Pattern Matching

1. **Exact Pattern Matching**
   - Pattern exactly match hona chahiye.
   - Example:

     Text = `"This is a book"`

     Pattern = `"book"` → Match found at position 11.
   - Algorithms: **Naive Search**, **Knuth-Morris-Pratt (KMP)**, **Boyer-Moore**.
2. **Approximate Pattern Matching**
   - Pattern me thoda mismatch allowed hota hai (spelling mistake, typo).
   - Example:

     Pattern = `"crickat"`

     Text contains `"cricket"` → Similar match found.
   - Algorithms: **Levenshtein Distance**, **Edit Distance**.

Applications in Information Retrieval

- **Keyword Search** in documents/webpages.
- **Spell Checking** suggestions.
- **Search Engines** me fast lookup.
- **DNA/Protein sequence matching** in bioinformatics.

Example (Exact Matching)

Text: `"I love programming in Python"`

Pattern: `"Python"`

- Match found at index position 23.

Structural Queries – Definition

Structural queries wo queries hoti hain jo data ke structure (format, tags, relationships) ko use karke search karti hain, sirf plain text content ko nahi. Ye un systems me use hoti hain jaha data structured form me hota hai, jaise XML, HTML, JSON, ya databases.

Key Points

- **Content ke saath-saath layout / structure bhi consider hota hai.**
- **Mostly markup languages (XML/HTML) ya hierarchical data pe kaam hoti hain.**
- **Advanced retrieval me help karti hain jaha sirf specific structure ka part chahiye hota hai.**

Example 1 – XML Search

Suppose hume XML file me sirf `<title>` tags ka content chahiye:

```xquery
//title
```

Ye query sirf *title elements* ka data return karegi, baaki ignore karegi.

Example 2 – HTML Search

Agar hume sirf web page ke `<h1>` headings chahiye:

```xpath
//h1
```

Example 3 – Database Structured Query

SQL query jo sirf table ke particular columns aur rows ko retrieve kare:

```sql
SELECT name, marks FROM students WHERE marks > 80;
```

Yaha **table ka structure** (columns, rows) use karke retrieval ho raha hai.

Advantages

- Precise results milte hain.
- Large structured datasets me fast retrieval.
- Content ke exact position/relationship ke basis pe search possible.

3. IR Models:
- Basic concepts
- Boolean Model
- Vector Model
- Probabilistic Model

c) Write a short note on probabilistic model. [4]

Explain various IR model in detail with their advantages & disadvantages.

c) Discuss vector model for information retrieval.

b) Compare and contrast the basic concepts, strengths, and limitations of the Boolean Model, Vector Model, and Probabilistic Model. Provide insights into when and why each model would be chosen to optimize search results, considering factors like ranking accuracy and complexity. **[8]**

1. Basic Concepts

- **Information Retrieval (IR):** The process of obtaining relevant information from a large repository (e.g., search engines).
- **Documents & Queries:** A document is any data item in the database; a query is the user's request for information.
- **Relevance:** The degree to which a document satisfies the user's information need.
- **Indexing:** Creating a data structure for fast document retrieval (e.g., inverted index).

Boolean Model – IR me kaise kaam karta hai

1. Kya hota hai?
Boolean Model ek **Information Retrieval technique** hai jisme documents aur queries ko **Boolean logic (AND, OR, NOT)** ke basis pe match kiya jata hai.
Ye sirf ye decide karta hai ki document **relevant hai ya nahi** – koi middle ground nahi.

Result → **Yes (1)** ya **No (0)**.

2. Representation kaise hota hai?

•**Document** → Terms ka set.

•**Query** → Boolean operators ke saath likha hua.

- **AND** → Dono terms present hone chahiye.
- **OR** → Koi ek term present ho to chalega.
- **NOT** → Ye term present nahi hona chahiye.

3. Example
Maan lo humare paas 3 documents hain:

Document	Content
D1	"AI and Machine Learning"
D2	"AI in Healthcare"
D3	"Machine Learning Basics"

Query: `AI AND Machine`

- D1 ✅ (AI bhi hai aur Machine bhi)
- D2 ❌ (Machine missing)
- D3 ❌ (AI missing)

Result → Sirf D1 milega.

4. Advantages (Fayde)

- Simple aur fast searching.
- Implementation easy.
- Exact match ke liye best.

---

5. Disadvantages (Nuksan)

- Koi ranking nahi → sab relevant documents equal treat hote hain.
- Strict matching → thoda sa miss ho gaya to result me nahi aayega.
- Complex queries normal users ke liye mushkil.

Vector Space Model – IR me kaise kaam karta hai

1. Kya hota hai?
Vector Space Model ek IR model hai jisme documents aur queries ko multi-dimensional space me vectors ke form me represent kiya jata hai.
Har dimension ek term ko represent karta hai.
Phir similarity measure karke decide hota hai ki kaunsa document query ke sabse close hai.

2. Kaam ka logic

1.Har document aur query ka vector banate hain.
2.Har term ka weight assign karte hain (**TF-IDF** use karke).
   1. **TF (Term Frequency)** → term kitni baar document me aaya.
   2. **IDF (Inverse Document Frequency)** → rare terms ko zyada importance.
3.Document aur query ke vectors ke beech **Cosine Similarity** nikalte hain.
4.Jo document ka similarity score zyada, wo zyada relevant.

3. Example

Query: `"AI Machine"`

- **D1:** "AI and Machine Learning"
- **D2:** "AI in Healthcare"
- **D3:** "Machine Learning Basics"

Sabka vector banta hai, similarity nikali jaati hai:

Doc	Similarity (cos θ)
D1	0.95
D2	0.60
D3	0.55

Result → D1 sabse relevant, fir D2, fir D3.

4. Advantages (Fayde)

- Ranking provide karta hai (sab equal nahi).
- Partial matching possible.
- Easy to extend with weights.

---

5. Disadvantages (Nuksan)

- Weight calculation me computation heavy.
- Large dataset me space requirement zyada.
- Similarity measure ka choice affect karta hai accuracy.

Probabilistic Model – IR me kaise kaam karta hai

1. Kya hota hai?
Probabilistic Model ek IR approach hai jo **ye predict karta hai ki ek document ke relevant hone ki probability kya hai** given a query.

Iska goal hota hai:
•**P(R|D, Q)** → Probability that document **D** is relevant (**R**) for query **Q**.

2. Kaam ka Logic

1. Shuru me ek **initial guess** lagate hain ki kaunse documents relevant ho sakte hain.
2. Har document ka **relevance probability** calculate karte hain.
3. Documents ko probability ke descending order me rank karte hain.
4. User feedback lene ke baad probability update karte hain (Relevance Feedback).

3. Common Model
Binary Independence Model (BIM):
•Assume karta hai ki terms independent hain.
•Har term ke relevant hone ki probability estimate karta hai.

- **Formula:**

```
Score(D) = Σ log [ (p_i / (1 - p_i)) / (q_i / (1 - q_i)) ]
```

Yahan `p_i` = probability of term in relevant docs, `q_i` = probability of term in non-relevant docs.

4. Example

Query: `"AI Machine"`

- System pehle kuch documents ko relevant maan leta hai (initial guess).
- Probability calculation ke baad pata chalta hai:
  - D1: 0.92
  - D2: 0.70
  - D3: 0.45
- Rank: D1 → D2 → D3.

---

5. Advantages (Fayde)

- Theoretical base strong (Probability theory).
- Ranking accurate hoti hai jab feedback milta hai.
- User feedback se continuously improve hota hai.

---

6. Disadvantages (Nuksan)

- Probability estimation mushkil hota hai (especially without feedback).
- Term independence assumption real life me sahi nahi hota.
- Implementation Boolean/Vector se complex hai.

Feature / Aspect	Boolean Model	Vector Space Model (VSM)	Probabilistic Model
Basic Concept	Documents aur queries ko Boolean logic (AND, OR, NOT) se match karta hai. Output binary hota hai – ya relevant ya nahi.	Documents aur queries ko vector form me represent karke similarity (Cosine) se rank karta hai.	Documents ka relevant hone ka probability nikalta hai given a query; uske hisaab se rank karta hai.
Ranking Accuracy	Koi ranking nahi, sirf match ya no match.	Ranking deta hai similarity score ke basis pe.	Ranking deta hai probability score ke basis pe (zyada accurate).
Strengths (Fayde)	Simple, fast, easy to implement.	Partial matches handle karta hai, better ranking deta hai.	Feedback se improve hota hai, accuracy high hoti hai.
Limitations (Nuksan)	Strict matching, partial relevant docs miss ho sakte hain, ranking nahi.	Computation heavy for large data, TF-IDF weight calculate karna padta hai.	Probability estimation mushkil, assumption ki terms independent hain.
When to Use (Kab use karein)	Jab exact match chahiye ho (legal docs, patents).	Jab ranking + partial match important ho (search engines).	Jab feedback mil sakta ho aur high accuracy chahiye (personalized search).
Complexity	Low	Medium	High

b) Explain exhaustivity & specificity with respect to index term weighting. **[6]**

Exhaustivity & Specificity in Index Term Weighting

Index Term Weighting ka matlab hota hai har term ko ek **weight** dena jo batata hai ki wo term kitna useful hai document ko represent karne me.
Isme do important concepts hote hain – **Exhaustivity** aur **Specificity**.

---

1. Exhaustivity

- **Meaning:** Ye measure karta hai ki ek document kitne **extent tak ek topic cover** karta hai.
- Agar document kisi topic ke saare aspects cover karta hai → **High Exhaustivity**.
- Agar sirf thoda part cover karta hai → **Low Exhaustivity**.
- **Example:**
  - Document on "Machine Learning" covering *history, algorithms, applications* → High Exhaustivity.
  - Document only on *linear regression* → Low Exhaustivity.

2. Specificity

- **Meaning:** Ye measure karta hai ki ek term ya document **kitna focused hai particular topic pe**.
- Agar term ka meaning clear hai aur irrelevant topics me use nahi hota → **High Specificity**.
- Agar term ka meaning vague hai aur multiple contexts me use hota hai → **Low Specificity**.
- **Example:**
  - Term “Neural Network” → High Specificity (AI me specific).
  - Term “Network” → Low Specificity (computer, social, transport, etc.).

Relation with Index Term Weighting

- **High Exhaustivity + High Specificity** → Term ka weight zyada hoga kyunki wo document ka content accurately represent karta hai.
- **Low Specificity** wale terms ka weight kam hoga kyunki wo ambiguous hote hain.
- Indexing me dono ka balance maintain karke accurate retrieval possible hota hai.