# ISR Oral

## 1️⃣ Difference between Data Retrieval and Information Retrieval

**Data Retrieval:**

- Focuses on **exact matching** of queries with data.
- Works with **structured data** (like databases).
- Example: SQL queries.
- Output is **precise and complete**.

**Information Retrieval:**

- Focuses on **relevant matching** of queries with documents.
- Works with **unstructured data** (like text or web pages).
- Example: Google Search.
- Output is **ranked by relevance**.

## 2️⃣ What is Conflation Algorithm? Explain its steps.

**Definition:**
A **Conflation Algorithm** is used to **merge different forms of the same word** into a single form (root/stem).
Example: *connect, connected, connecting* → *connect*

**Steps:**

1. **Tokenization:**
    - Break the text into words or tokens.
    - Example: "He is connecting devices" → [he, is, connecting, devices]
2. **Normalization:**
    - Convert all words to lower case and remove symbols or numbers.
    - Example: "Connecting" → "connecting"
3. **Stopword Removal:**
    - Remove common words like *is, the, and, of* that don't add meaning.
    - Example: [he, is, connecting, devices] → [connecting, devices]
4. **Stemming / Lemmatization:**
    - Cut off prefixes or suffixes to find the **root word**.
    - Example: *connected, connecting, connection* → *connect*
5. **Grouping (Conflation):**

- Group all similar word forms under one root word.
- Example: {connected, connecting, connection} → **connect**

6. **Indexing:**
   - Store these root forms in the index for **easy and fast searching**.

---

**Example:**

Input: "Students are studying and studied well."
After Conflation → [student, study, well]

**Purpose / Advantages**

- Improves **recall** by matching related terms.
- Reduces **index size** by storing only root forms.
- Enhances **retrieval efficiency** in IR systems.

### 3️⃣ What is Luhn's Idea? Explain the sections in it.

**Definition:**
Luhn's Idea (by Hans Peter Luhn) proposed that **significant words** in a document can represent its **content or topic**.

**Sections:**

1. **Frequency Analysis:**
   - Count how many times each word appears.
2. **Selection of Significant Words:**
   - Ignore high-frequency (common) and very low-frequency (rare) words.
3. **Automatic Abstracting / Indexing:**
   - Use significant words to **summarize or index** the document.

**Example:**

Document:
*"Neural networks are used in machine learning. Neural networks help in pattern recognition."*

1. **Frequency Analysis:**
   - neural (2), networks (2), machine (1), learning (1), help (1), pattern (1), recognition (1)
2. **Significant Words:**
   - neural, networks, machine, learning, pattern, recognition
3. **Summary / Index (Luhn's Idea):**
   - **"Neural networks, machine learning, pattern recognition"**

---

✅ **Simple Oral Tip:**

"Luhn's Idea finds important words in a document by counting frequency, ignoring too common or rare words, and then uses them to summarize or index the document."

### 4️⃣ What are Stopwords?

**Definition:**
Stopwords are **common words** that do **not carry significant meaning** and are **removed** during text processing.

**Examples:**
"the", "is", "and", "of", "in", "to"

**Purpose:**
Removing them reduces noise and improves retrieval efficiency.

### 5️⃣ What is a Document Representative?

**Definition:**
A **Document Representative** is a **simplified representation** of a document used for indexing and retrieval.

**Examples:**

- Set of keywords
- Weighted term vector (TF-IDF)
- Summary or concept list

**Purpose:**
To capture the **essence or meaning** of the document for faster searching.

TF-IDF stands for **Term Frequency–Inverse Document Frequency**. It's a way to measure how important a word is in a document compared to a collection (or corpus) of documents. It's widely used in information retrieval, search engines, and text analysis. Let me break it down simply:

---

### 1️⃣ Term Frequency (TF)

This measures how often a word appears in a document.

$$\text{TF(word)} = \frac{\text{Number of times word appears in the document}}{\text{Total number of words in the document}}$$

**Example:**
If the word "data" appears 3 times in a document of 100 words:

$$TF(\text{"data"}) = 3/100 = 0.03$$

---

### 2 Inverse Document Frequency (IDF)

This measures how unique or rare a word is across all documents.

$$\text{IDF(word)} = \log \frac{\text{Total number of documents}}{\text{Number of documents containing the word}}$$

- Common words like "the" or "is" appear in almost every document → low IDF.
- Rare words like "cryptography" appear in fewer documents → high IDF.

---

### 3 TF-IDF

Multiply TF and IDF to get the importance of a word in a document:

$$\text{TF-IDF} = TF \times IDF$$

**Purpose:**

- High TF-IDF → Word is frequent in this document but rare across other documents → important keyword.
- Low TF-IDF → Word is either common everywhere or rare in this document → less important.

**Example:**

- "Data" occurs frequently in one document (high TF) but also in many documents (low IDF) → medium importance.
- "TF-IDF" occurs rarely in most documents but frequently in one → very important in that document.

### 6 Explain Indexing, Exhaustivity, and Specificity.

**Indexing:**
Process of **organizing documents** by their important terms to enable quick searching.

**Exhaustivity:**

- Degree to which **all topics or concepts** in a document are covered by the index terms.
- High exhaustivity = more terms.

**Specificity:**

- Degree to which the **index terms are precise** to the subject.
- High specificity = more focused terms.

**Example:**
Document on "Neural Networks in Healthcare"

- High exhaustivity → "AI", "Neural Networks", "Healthcare", "Diagnosis"
- High specificity → "Deep Learning for Medical Imaging"

## 7️⃣ Five commonly used measures of association in Information Retrieval

1. **Jaccard Coefficient**
   → Measures intersection over union of terms.
2. **Cosine Similarity**
   → Measures angle between two term vectors.
3. **Dice Coefficient**
   → Similar to Jaccard but gives more weight to overlap.
4. **Pearson's Correlation Coefficient**
   → Measures linear relationship between variables.
5. **Simple Matching Coefficient (SMC)**
   → Ratio of matching attributes (both 1s and 0s) to total attributes.

## 1️⃣ Jaccard Coefficient

**Definition:**
Measures the similarity between two sets by dividing the size of their intersection by the size of their union.

$$\text{Jaccard}(A, B) = \frac{|\ A \cap B\ |}{|\ A \cup B\ |}$$

**Explanation:**

- It is used when we want to see how many terms two documents share relative to the total unique terms in both.
- Value ranges from 0 (no common terms) to 1 (all terms are common).

**Example:**

- Document 1 terms: {data, science, AI}
- Document 2 terms: {AI, machine, learning}
- Intersection = {AI} → 1
- Union = {data, science, AI, machine, learning} → 5

$$\text{Jaccard} = 1/5 = 0.2$$

---

## 2️⃣ Cosine Similarity

**Definition:**
Measures similarity as the **cosine of the angle** between two term vectors in a multi-dimensional space.

$$\text{Cosine}(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

**Explanation:**

- Treat each document as a vector of term weights (e.g., TF-IDF).
- Cosine similarity ignores document length and focuses on **direction**.
- Value ranges from 0 (orthogonal, no similarity) to 1 (identical).

**Example:**

- Document vectors: A = [1, 2, 0], B = [1, 1, 0]

$$\text{Cosine} = \frac{1*1+2*1+0*0}{\sqrt{1^2+2^2+0^2}*\sqrt{1^2+1^2+0^2}} = \frac{3}{\sqrt{5}*\sqrt{2}} \approx 0.949$$

---

### 3 Dice Coefficient

**Definition:**
Similar to Jaccard but **gives more weight to the common terms**.

$$\text{Dice}(A, B) = \frac{2 \mid A \cap B \mid}{\mid A \mid + \mid B \mid}$$

**Explanation:**

- Good for small sets where overlaps are important.
- Value ranges from 0 (no similarity) to 1 (perfect match).

**Example:**

- Document 1 terms: {data, science, AI} → 3
- Document 2 terms: {AI, machine, learning} → 3
- Intersection = 1

$$\text{Dice} = \frac{2*1}{3+3} = \frac{2}{6} = 0.333$$

---

### 4 Pearson's Correlation Coefficient

**Definition:**
Measures the **linear relationship** between two variables or term vectors.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

**Explanation:**

- Used when comparing the **patterns of term occurrences** rather than exact matches.
- Value ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation).
- 0 means no linear relationship.

**Example:**

- Document term counts: X = [1, 2, 0], Y = [2, 4, 0] → perfectly proportional → r = 1

---

### 5 Simple Matching Coefficient (SMC)

**Definition:**
Measures similarity as the **ratio of matching attributes** (both 1s and 0s) to total attributes.

$$\text{SMC} = \frac{M_{11} + M_{00}}{M_{11} + M_{10} + M_{01} + M_{00}}$$

Where:

- $M_{11}$= count of attributes where both are 1
- $M_{00}$= count of attributes where both are 0
- $M_{10}$, $M_{01}$= mismatches

**Explanation:**

- Good for **binary term vectors** (presence/absence of words).
- Value ranges from 0 (completely different) to 1 (identical).

**Example:**

- Document 1 vector: [1, 0, 1, 0]
- Document 2 vector: [1, 0, 0, 0]
- Matches: $M_{11} = 1, M_{00} = 2$

$$\text{SMC} = \frac{1 + 2}{4} = 0.75$$

### 8 Why normalized versions of the Simple Matching Coefficient are used for measures of association?

**Answer:**
Because the **raw matching count** can be misleading when documents or queries have different lengths.
Normalization ensures that:

- The **value lies between 0 and 1**,
- The measure is **independent of document size**,
- It gives **fair comparison** across documents.

ASSIGNMENT-2

## 1 What is Clustering?

**Definition:**
Clustering is the process of **grouping similar objects or documents** into clusters so that objects in the same cluster are **more similar to each other** than to objects in other clusters.

**Example:**
Documents about *sports* go in one cluster, *politics* in another.

## 2 Types of Clustering

1. **Hierarchical Clustering:**
    - Builds a **tree of clusters**.
    - Can be **agglomerative** (start with single items → merge) or **divisive** (start with all items → split).
2. **Partitioning Clustering:**
    - Divides objects into a **fixed number of clusters** (like K-means).
3. **Density-Based Clustering:**
    - Forms clusters based on **dense regions** in data (like DBSCAN).
4. **Single-Pass / Incremental Clustering:**
    - Clusters are created **one by one** as data comes in.

## 3 Single Pass Clustering Algorithm

**Steps:**

1. Take the **first document** → create the first cluster.
2. Take the **next document** → compute similarity with existing clusters.
3. If similarity > threshold → **add to that cluster**.
4. If similarity < threshold → **create a new cluster**.
5. Repeat for all documents.

**Example:**

- Doc1 → Cluster1

- Doc2 similar to Doc1 → Cluster1
- Doc3 very different → Cluster2

### 4 Clustering Using Similarity Measures

**Definition:**
Clustering can be done by computing **similarity between documents**.

**Common similarity measures:**

- **Cosine similarity** → angle between vectors
- **Jaccard coefficient** → intersection / union of terms
- **Euclidean distance** → closeness in numeric space

**Example:**
Doc1: ["apple", "banana"]
Doc2: ["apple", "orange"]
Cosine similarity can tell how close they are → cluster accordingly.

### 5 IR Models

**Definition:** Models define **how queries and documents are matched** in Information Retrieval.

1. **Boolean Model:** Documents either match or not.
2. **Vector Space Model:** Documents represented as vectors; similarity is calculated.
3. **Probabilistic Model:** Ranks documents based on probability of relevance.
4. **Language Model:** Uses probability of generating query from document.

### 6 Boolean Search

**Definition:**
A search technique that uses **AND, OR, NOT** operators.

**Example:**

- Query: "machine AND learning" → Documents containing **both words**
- Query: "apple OR orange" → Documents containing **either word**
- Query: "banana NOT apple" → Documents containing **banana but not apple**

---

### 7 Multi-Pass Clustering Technique

**Definition:**

- Uses **more than one pass** over the data to improve clustering.
- First pass: assign documents to **rough clusters**.

- Second pass: **refine clusters** for better accuracy.

**Use:** When data is **too large for single-pass clustering**.

**Steps / How It Works:**

1. **First Pass – Rough Clustering**
    - Go through all documents **once**.
    - Assign each document to a cluster based on a **similarity threshold**.
    - Some clusters may not be perfect; some documents may be misclassified.
    - Goal: **quick initial grouping**.
2. **Second Pass – Refinement**
    - Go through all clusters and documents **again**.
    - Recompute **similarity of documents to clusters**.
    - Move documents to a **more appropriate cluster** if similarity is higher.
    - May also **merge or split clusters** for better accuracy.
3. **Additional Passes (Optional)**
    - For very large or complex datasets, more passes may be done.
    - Each pass improves **cluster quality** and **reduces errors**.

### 8️⃣ Clustering Using Dissimilarity Matrix

**Definition:**

- Create a **matrix showing dissimilarity** (distance) between every pair of documents.
- Cluster documents that are **closest together**.

**Effect of Threshold:**

- **High threshold** → fewer clusters, more documents in each cluster
- **Low threshold** → more clusters, smaller clusters

**Example:**

| **Doc** | **Doc1** | **Doc2** | **Doc3** |
|---|---|---|---|
| Doc1 | 0 | 0.2 | 0.8 |
| Doc2 | 0.2 | 0 | 0.7 |
| Doc3 | 0.8 | 0.7 | 0 |

- Threshold = 0.5 → Doc1 & Doc2 together, Doc3 separate

### 9️⃣ K-list

**Definition:**

- A **list of clusters** for each document **sorted by similarity**.
- Helps in **cluster-based retrieval** and quick access to related documents.

**Example:**

- Doc1 → Cluster1 (0.9), Cluster2 (0.6)
- Doc2 → Cluster2 (0.8), Cluster1 (0.5)

---

### 🔟 Cluster-Based Retrieval

**Definition:**

- Instead of retrieving single documents, **retrieve entire clusters** of related documents.
- Helps users **find groups of relevant documents** quickly.

**Example:**
Query: "Machine Learning"

- Retrieves cluster containing all documents about *ML algorithms, datasets, and applications*

| Feature | Single-Pass Clustering | Multi-Pass Clustering |
|---|---|---|
| **Definition** | Processes each document **once** and assigns it to a cluster immediately. | Processes documents **multiple times** to refine clusters. |
| **Cluster Quality** | Clusters may be **rough or inaccurate**, especially for large datasets. | Clusters are **more accurate and refined** due to multiple passes. |
| **Processing** | **One pass** through the data. | **Two or more passes** over the data. |
| **Complexity** | Simple, **faster**, but less accurate. | Slower, but produces **better-quality clusters**. |
| **Use Case** | Small datasets or when speed is important. | Large or complex datasets where accuracy matters. |
| **Example** | Doc assigned to first cluster it is similar to; may not be perfect. | First pass forms rough clusters; second pass rechecks similarity and reassigns documents. |

**Assignment No :3**

## 1 What are Inverted Files?

**Definition:**
An **inverted file** (or inverted index) is a **data structure** that maps **words (terms) to the documents** that contain them.

- Helps **fast searching** in information retrieval.

**Example:**

- Doc1: "Machine learning is fun"
- Doc2: "Deep learning uses neural networks"
  **Inverted File:**

learning → Doc1, Doc2

machine → Doc1

deep → Doc2

neural → Doc2

## 2 What is Indexing?

**Definition:**
Indexing is the process of **creating a data structure** (like inverted files) to allow **quick search and retrieval** of documents.

**Example:**
Instead of scanning all documents, search uses the **index** to find documents containing the query word.

## 3 What is Vocabulary and Occurrences?

- **Vocabulary:**
  Set of **unique terms** in the collection.
  - Example: In Doc1 & Doc2 → Vocabulary = {machine, learning, fun, deep, uses, neural, networks}
- **Occurrences (Postings):**
  List of **documents where each term appears**.
  - Example: learning → Doc1, Doc2

## 4 How Search is Carried Out on Inverted Index?

**Steps:**

1. Convert query into terms (tokenization).

2. Look up each term in the **inverted index**.
3. Get **list of documents (postings)** for each term.
4. Combine results using **Boolean operators (AND, OR, NOT)**.
5. Return **relevant documents**.

**Example:**
Query: learning AND neural → Doc2

### 5 How to Index Multimedia Objects?

- Convert multimedia into **representative features**:
  - Images → colors, shapes, textures
  - Videos → keyframes, motion vectors
  - Audio → frequency, pitch
- Store these features in an **index** for fast search.
- Use **content-based retrieval** instead of text-based.

### 6 Limitations of Inverted Index

1. Large storage for huge document collections.
2. Updates (insert/delete) are **costly**.
3. Cannot handle **proximity** or semantic meaning easily.
4. Complex for **multimedia or unstructured data**.

### 7 What is Suffix-Array and Suffix-Tree?

- **Suffix-Array:** Sorted array of all suffixes of a text.
  - Example: Text = "banana" → suffixes: banana, anana, nana, ana, na, a → sorted array: a, ana, anana, banana, na, nana
- **Suffix-Tree:** Tree structure containing all **suffixes of a text**.
  - Helps in **fast substring search**.

### 8 What is the Concept of Signature Files?

**Definition (Simple)**

A **signature file** is like a **shortcut or fingerprint** for a document. Instead of checking the whole document when searching, we first compare these fingerprints. If the fingerprint matches the query, then we look at the full document.

---

**How it Works**

1. **Document → Signature**
    - Each document is turned into a **fixed-length bit string** (like 16 bits or 32 bits).
    - This is done using a **hash function** that compresses the document's content.
2. **Query → Signature**
    - The search query is also converted into a signature using the same method.
3. **Filter Candidates**
    - Compare the query signature with document signatures.
    - Only documents with **matching signatures** are checked fully.

---

**Example**

- Document: "AI is the future" → hashed → 10110011 (16-bit signature)
- Query: "AI future" → hashed → 10110010
- Compare signatures: If they match, we check the actual document. If not, we skip it.

**Purpose:**

- Makes search **faster** by ignoring documents that definitely don't match.
- Saves time and storage compared to scanning full documents every time.

### 9️⃣ Working of Inverted Files

**Steps:**

1. Scan all documents → extract terms (tokenization).
2. Remove **stopwords** and apply **stemming**.
3. Create **vocabulary** (unique terms).
4. Build **postings list** (document IDs for each term).
5. Store as **inverted index**.

---

### 🔟 Applications of Inverted Index

1. Web search engines (Google, Bing)
2. Digital libraries
3. Document management systems
4. Question-answering systems

5. Text mining and NLP applications

---

### 1️⃣1️⃣ Working of Signature Files

**Steps:**

1. Compute **signature (bit-vector)** for each document.
2. Store in **signature file**.
3. For a query, compute its **signature**.
4. Compare query signature with document signatures → get **candidate documents**.
5. Verify candidates with **actual content**.

**Example:**

- Doc signature: 10101100
- Query signature: 10100100 → match → check document

## Assignment-4

### 1️⃣ What is Precision and Recall in IR System?

- **Precision:**
  Measures the **accuracy** of retrieved documents.
  - Formula: **Precision = (Number of Relevant Documents Retrieved) ÷ (Total Documents Retrieved)**
  - Example: Retrieved 10 docs, 7 are relevant → Precision = 7/10 = 0.7
- **Recall:**
  Measures the **coverage** of relevant documents retrieved.
  - Formula: **Recall = (Number of Relevant Documents Retrieved) ÷ (Total Relevant Documents in Collection)**
  - Example: Total relevant docs = 20, retrieved 7 → Recall = 7/20 = 0.35

### 1️⃣ Precision

- **What it tells:** How **accurate** your search results are.
- **In other words:** Of all the documents you found, how many were actually useful?

$$\text{Precision} = \frac{\text{Relevant documents retrieved}}{\text{Total documents retrieved}}$$

**Example:**

- You search and get 10 documents.

- 7 of them are actually useful (relevant).

$$\text{Precision} = 7/10 = 0.7\ (70\%)$$

✅ High precision → Most retrieved documents are relevant.

---

### 2️⃣ Recall

- **What it tells:** How **complete** your search is.
- **In other words:** Of all the useful documents out there, how many did you actually find?

$$\text{Recall} = \frac{\text{Relevant documents retrieved}}{\text{Total relevant documents in the collection}}$$

**Example:**

- There are 20 relevant documents in the whole collection.
- Your search found 7 of them.

$$\text{Recall} = 7/20 = 0.35\ (35\%)$$

✅ High recall → You found most of the relevant documents.

**Precision and Recall using TP, FP, FN**

| Measure | Formula | Explanation |
|---|---|---|
| Precision | $\text{Precision} = \frac{TP}{TP+FP}$ | TP = relevant documents retrieved, FP = non-relevant documents retrieved |
| Recall | $\text{Recall} = \frac{TP}{TP+FN}$ | TP = relevant documents retrieved, FN = relevant documents not retrieved |

---

**Easy way to remember:**

- **Precision = "How correct are my results?"**
- **Recall = "Did I find everything I should?"**

### 2️⃣ What is Relevance of Document?

- **Definition:**
  A document is **relevant** if it **satisfies the user's information need** or query.
- **Example:** Query: "Machine Learning Basics"
  - Document about *ML algorithms* → Relevant

- Document about *Python programming unrelated to ML* → Not Relevant

### 3 What are the Metrics to Measure Information Systems?

Common metrics:

1. **Precision** → Accuracy of retrieved documents
2. **Recall** → Coverage of relevant documents
3. **F-Measure / F1-Score** → Harmonic mean of precision and recall
   - Formula: F1 = 2 × (Precision × Recall) ÷ (Precision + Recall)
4. **Accuracy** → Fraction of correct results (relevant + non-relevant)
5. **Mean Average Precision (MAP)** → Average precision across multiple queries

### 5 What is the Problem with These Two Measures?

- **Precision vs Recall Conflict:**
  - High precision → may retrieve **fewer documents**, missing many relevant ones
  - High recall → may retrieve **more documents**, including non-relevant ones
- They do not provide a **single comprehensive measure** of retrieval performance.

---

### 6 What is Precision-Recall Trade-off?

- **Definition:**
  Increasing **recall** usually **decreases precision**, and increasing **precision** usually **decreases recall**.
- **Explanation:**
  - Retrieving **more documents** improves recall but adds irrelevant ones → lowers precision
  - Retrieving **fewer, highly relevant documents** improves precision but may miss some → lowers recall

**Example:**

- Query: "Neural Networks"
  - Retrieve top 5 docs → 5 relevant → Precision = 1, Recall = 0.5
  - Retrieve top 10 docs → 7 relevant → Precision = 0.7, Recall = 0.7

The trade-off is often visualized as a **Precision-Recall curve**.

**Assignment-5**

## 1 What is Harmonic Mean (F-measure) and E-measure in IR systems?

- **F-measure (F1-Score):**
  - Combines **Precision** and **Recall** into a **single measure**.
  - It is the **harmonic mean** of precision and recall.
  - Gives **balanced importance** to both precision and recall.
- **E-measure (Error Measure):**
  - Combines precision and recall into a **single error-based metric**.
  - Penalizes **low precision or low recall** more than F-measure.
  - Defined as:

$$E = 1 - F_\beta$$

(or a variant depending on weight)

## 2 How are F-measure and E-measure calculated? (Formulae)

- F-measure (F1-score):

$$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- General Fβ-measure:

$$F_\beta = (1 + \beta^2)\frac{P \cdot R}{(\beta^2 \cdot P) + R}$$

- E-measure:

$$E = 1 - F_\beta$$

Example:

- Precision = 0.7, Recall = 0.35

$$F = 2 \times \frac{0.7 \times 0.35}{0.7 + 0.35} = 0.467$$

$$E = 1 - (\downarrow 7 = 0.533$$

## 3️⃣ Difference between F-measure and E-measure

| Feature | F-measure | E-measure |
|---|---|---|
| Purpose | Combines precision & recall | Represents **error / penalty** |
| Range | 0 to 1 (higher is better) | 0 to 1 (lower is better) |
| Interpretation | Higher value → better performance | Lower value → better performance |
| Formula | Harmonic mean of P & R | $E = 1 - F\beta$ |

## 4️⃣ Metrics to Measure Information Systems

1. **Precision** → Accuracy of retrieved docs
2. **Recall** → Coverage of relevant docs
3. **F-measure / F1-score** → Combines precision & recall
4. **E-measure** → Error-based metric
5. **Accuracy** → Fraction of correct results
6. **MAP (Mean Average Precision)** → Average precision across queries

## 5️⃣ Advantages of F-measure and E-measure

- **F-measure:**
  - Gives **single score** for precision & recall
  - Easy to **compare IR systems**
  - Balances **precision vs recall**
- **E-measure:**
  - Highlights **error** in retrieval
  - Penalizes **systems with very low precision or recall**
  - Useful in **performance evaluation**

**Assignment-6**

---

## 1️⃣ What is Extraction (or Feature Extraction)?

**Definition:**
Feature Extraction is the process of **identifying and isolating important attributes or characteristics** from raw data (like images, audio, or text) so that they can be **used for indexing, retrieval, or analysis**.

**Example:**

- For an image of a flower → features could be **color, shape, texture**.

---

## 2️⃣ How Images are Indexed?

Images are indexed based on their **features**:

1. **Extract features** like color, shape, texture.
2. **Convert features into a vector or numerical representation**.
3. **Store in an index** (like an inverted index or feature database).
4. During search, **query image features** are compared with indexed vectors.

---

## 3️⃣ How Color is Extracted from an Image

- Convert the image into a **color space** (RGB, HSV, etc.).
- Compute **color histograms** → count of pixels for each color.
- Represent the image using **dominant colors or histogram bins**.

**Example:**

- Image with 70% red, 20% green, 10% blue → Feature vector = [0.7, 0.2, 0.1]

---

## 4️⃣ What is Multimedia IR? Steps for Data Retrieval

**Definition:**
Multimedia Information Retrieval (MIR) is retrieving **images, audio, video** based on their **content features**, instead of text.

**Steps:**

1. **Content Representation:** Extract features (color, shape, texture, audio frequency).
2. **Indexing:** Store feature vectors in a database.
3. **Query Processing:** Extract features from query object (image/audio/video).
4. **Similarity Computation:** Compare query features with stored features.
5. **Ranking & Retrieval:** Return results **sorted by relevance**.

---

### 5 Use of Image Features

- Helps in **searching and retrieving images**.
- Reduces **storage** by storing only features instead of full content.
- Improves **accuracy** in image-based queries.

---

### 6 Features of Image and Applications

| Feature | Example / Application |
|---|---|
| Color | Content-based image search |
| Shape | Object recognition, medical imaging |
| Texture | Satellite imagery, pattern recognition |
| Edge/Contour | Face recognition, fingerprint analysis |

---

### 7 How to Compare Two Images and Calculate Relevancy

1. Represent each image as a **feature vector**.
2. Compute **similarity** between vectors:
    - **Euclidean distance** → distance between vectors
    - **Cosine similarity** → angle between vectors
    - **Histogram intersection** → for color features
3. Rank images based on **similarity score** → higher similarity = more relevant

**Example:**

- Query image feature = [0.7, 0.2, 0.1]
- Database image feature = [0.6, 0.25, 0.15]
- Euclidean distance = small → highly relevant

---

### 8 Applications of Feature Extraction

- **Content-based Image Retrieval (CBIR)**
- **Face recognition and biometric systems**
- **Medical image analysis**
- **Video retrieval and surveillance**
- **Object recognition in AI systems**

- **Satellite and remote sensing image analysis**

---

**Assignment–7**

---

### 1️⃣ What are Search Engines? Name a few of them.

**Definition:**
A **Search Engine** is a software system that helps users **find information on the web** by searching keywords and returning relevant web pages.

**Examples:** Google, Bing, Yahoo, DuckDuckGo, Baidu.

---

### 2️⃣ How Search Engine Works

Search engines work in **three main steps**:

1. **Crawling:** Collect web pages using programs called crawlers or spiders.
2. **Indexing:** Analyze and store page content in a huge database.
3. **Retrieval/Ranking:** When a user searches, the engine retrieves matching pages and **ranks them** by relevance using algorithms like PageRank.

---

### 3️⃣ What is Web Crawling?

**Definition:**
Web Crawling is the process of **automatically browsing** web pages and collecting their data for indexing.

**Tool Used:**
A **Web Crawler (Spider or Bot)** fetches pages and follows links to discover new ones.

---

### 4️⃣ What is Robot Exclusion Protocol (robots.txt)?

**Definition:**
A **robots.txt** file tells web crawlers **which pages or sections** of a website they are **not allowed to crawl**.

**Example:**

User-agent: *

Disallow: /admin/

👉 This means all crawlers are blocked from the **/admin/** section.

---

### 5️⃣ Significance of robots.txt

- Prevents **overloading** the server with crawler requests.
- Keeps **private or duplicate pages** hidden from search engines.
- Helps maintain **site security and efficiency**.

---

### 6 Strategies Used by Crawlers

1. **Breadth-First Search (BFS):** Crawl links level by level.
2. **Depth-First Search (DFS):** Crawl deep into each link before moving to the next.
3. **Focused Crawling:** Crawl only pages related to a specific topic.
4. **Parallel Crawling:** Use multiple crawlers at the same time for speed.
5. **Adaptive Crawling:** Adjust strategy based on page importance and freshness.

---

### 7 What is PageRank?

**Definition:**
PageRank is an **algorithm developed by Google** to rank web pages based on their **importance and link structure**.

**Idea:**
Pages linked by many other important pages have **higher rank**.

**Formula (simplified):**
[
PR(A) = (1 - d) + d \times [PR(B1)/L(B1) + PR(B2)/L(B2) + ...]
]
where **d** = dampening factor (usually 0.85).

---

### 8 Significance of Dampening Factor

- The **Dampening Factor (d)** prevents infinite loops in the link structure.
- It represents the **probability that a user continues clicking links**.
- A typical value is **0.85** — meaning 85% chance the user keeps clicking, 15% chance they stop.

---

### 9 Crawler Architectures

1. **Basic Architecture:** Single crawler that downloads and stores pages.
2. **Parallel / Distributed Architecture:** Multiple crawlers work together to speed up the process.
3. **Focused Crawler:** Targets specific topics or domains.
4. **Incremental Crawler:** Updates only changed or new pages.

---

### 🔟 Harvest Architecture

**Definition:**
The **Harvest architecture** is a framework for gathering and indexing internet data.

**Components:**

1. **Gatherer:** Collects information from different sites.
2. **Broker:** Indexes and provides search interface.

**Working:**

- Gatherers collect data → send to Brokers → Brokers create searchable indexes for users.

---

### 1️1️ Working of Google Crawler

1. **Crawling:** Googlebot scans web pages using sitemaps and links.
2. **Indexing:** Analyzes content, keywords, and meta tags to store information.
3. **Ranking:** Uses **PageRank** and other algorithms to show the most relevant results.
4. **Updating:** Constantly revisits pages to keep the index fresh.

---

### 1️2️ Challenges Involved in Searching the Web

1. **Huge Size of the Web** → Billions of pages.
2. **Dynamic Content** → Pages change or disappear frequently.
3. **Duplicate Pages** → Many pages have the same content.
4. **Multimedia and Non-Text Data** → Hard to index images/videos.
5. **Spam and Low-Quality Content** → Affects ranking quality.
6. **Scalability and Storage Issues** → Need large servers and fast processing.

---

## ⛈️ ASSIGNMENT-8 – APIs

---

### 1️ What are APIs and their Use?

**Definition:**
API stands for **Application Programming Interface**.
It allows two software applications to **communicate or share data** with each other.

**Use:**

- Connects your app to other services.

- Helps **fetch or send data** (e.g., weather, maps, payment).
- Saves time — you don't need to build everything from scratch.

**Example:**
Using the **OpenWeatherMap API** to get live weather details for a city.

---

### 2️⃣ How to Use an API?

1. **Register** on the API provider's website (get an **API key**).
2. **Send a request** to the API URL (using methods like GET or POST).
3. **API responds** with data (usually in **JSON** format).
4. **Use the data** in your application (like showing weather, temperature, etc.).

**Example:**

https://api.openweathermap.org/data/2.5/weather?q=Pune&appid=YOUR_API_KEY

---

### 3️⃣ Which API You Have Used in Assignment-8?

✅ We have used the **OpenWeatherMap API**.

---

### 4️⃣ Explain the API You Used (OpenWeatherMap API)

**Name:** OpenWeatherMap API
**Purpose:** To provide **real-time weather data** for any city.
**Data Returned:** Temperature, humidity, wind speed, weather condition, etc.
**Steps Used:**

1. Collected API key from OpenWeatherMap.
2. Fetched data using the API link and **city name** as input.
3. Displayed the weather details in a **web or app interface**.

**Example Output:**
→ "Pune: 30°C, Clear Sky, Humidity 45%"

---

## 📘 ASSIGNMENT-9 – Case Study & Recommendation Systems

---

### 1️⃣ What is Case Study?

**Definition:**
A Case Study is a **detailed examination of a specific system, project, or topic** to understand how it works and what can be improved.

---

### 2️⃣ On Which Topic You Have Done Case Study?

✅ Example Answer:

"I have done my case study on **Recommendation Systems** and how they personalize content for users."

(You can adjust if your actual topic differs.)

---

### 3️⃣ What are Recommendation Systems?

**Definition:**
Recommendation Systems are **AI-based systems** that suggest items (like products, movies, or documents) based on **user behavior and preferences**.

**Examples:**

- Netflix recommending movies
- Amazon suggesting products
- Spotify recommending songs

---

### 4️⃣ How Are Recommendation Systems Classified (Types)?

| Type | Description | Example |
|---|---|---|
| **1. Content-Based Filtering** | Recommends items similar to what the user liked before. | Netflix showing movies of the same genre you watched. |
| **2. Collaborative Filtering** | Recommends items based on what **similar users** liked. | "Users who bought this also bought…" on Amazon. |
| **3. Hybrid Filtering** | Combines both content-based and collaborative methods. | YouTube recommendations. |

---

### 5️⃣ Collaborative Filtering Recommendation

**Idea:**
Users with **similar interests** get **similar recommendations**.

**Steps:**

1. Identify users with similar preferences.
2. Recommend items liked by similar users.

**Example:**
If Ved and Omkar both liked "Java tutorial," and Omkar liked "React tutorial," the system recommends "React tutorial" to Ved.

---

### 6️⃣ Content-Based Filtering Recommendation

**Idea:**
Recommends items that are **similar in content** to those the user already liked.

**Steps:**

1. Analyze item features (genre, keywords, description).
2. Match with the user's history or profile.

**Example:**
If you liked the movie *Iron Man*, the system recommends other **action or superhero** movies.