CBCS SCHEME

USN | | | | | | | | | | |

BAI/BCY515B

Fifth Semester B.E./B.Tech. Degree Examination, Dec.2025/Jan.2026
Information Retrieval

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Define Information Retrieval. Explain information retrieval in libraries and digital libraries.	8	L2	CO1
	b.	Explain Software Architecture of the IR system.	12	L2	CO1
		OR			
Q.2	a.	Explain classic versus dynamic model of information seeking.	10	L2	CO1
	b.	Explain design and evaluation of search interfaces.	10	L2	CO1
		Module – 2			
Q.3	a.	Write the characteristics of IR Model.	8	L2	CO2
	b.	Explain TF-IDF weights.	12	L2	CO2
		OR			
Q.4	a.	Explain Bayesian Network Models.	10	L2	CO2
	b.	Explain Extended Boolean Model.	10	L2	CO2
		Module – 3			
Q.5	a.	List and explain evaluation measures in TREC collections.	10	L2	CO3
	b.	Explain explicit relevance feedback information.	10	L2	CO3
		OR			
Q.6	a.	Explain document preprocessing text operations (transformations).	10	L2	CO3
	b.	Explain organizing documents.	10	L2	CO3
		Module – 4			
Q.7	a.	Explain simple string Horspool's algorithm.	10	L2	CO4
	b.	Explain construction of suffix arrays for large texts.	10	L2	CO4
		OR			
Q.8	a.	Explain Multiple Word Queries.	10	L2	CO4
	b.	Explain structure for Tries and Suffix Trees.	10	L2	CO4
		Module – 5			
Q.9	a.	Explain Cluster-Based Architecture.	10	L2	CO5
	b.	Explain Retrieval Task.	10	L2	CO5
		OR			
Q.10	a.	Explain Static Vs Dynamic Structure.	10	L2	CO5
	b.	Explain Signal Hierarchy Vs Multiple Hierarchy.	10	L2	CO5

* * * * *

IR ANSWER KEY (DEC – Jan 25 -26)

MODULE -1

1a. Defiine Information Retrival. Explain information retrival in libraries and digital libraries

Definition of Information Retrieval (IR):

Information Retrieval (IR) is the process of identifying, searching, and retrieving relevant information objects such as documents, books, articles, or web pages from a large collection, in response to a user's information need expressed as a query. The main objective of IR is to retrieve useful and relevant information rather than exact data

Information Retrieval in Libraries:

Traditional libraries have long used IR techniques to help users locate information efficiently. The key aspects are:

1. Catalog-Based Retrieval:
   Libraries maintain catalogues (card catalogs or OPACs) that store metadata such as author, title, subject, and classification number.
2. Controlled Vocabulary:
   Standard classification systems like Dewey Decimal Classification (DDC) and Library of Congress Classification (LCC) are used to organize documents.
3. Manual and Semi-Automated Search:
   Users search using keywords, author names, or subject headings, often with the help of librarians.
4. Limited Collection Size:
   Library collections are relatively smaller and well-structured, making retrieval more precise but less flexible.
5. Physical Access:
   Retrieved information usually requires physical access to books or journals.

Information Retrieval in Digital Libraries:

Digital libraries extend traditional IR concepts using computer-based and networked technologies. Their main features include:

1. Electronic Document Collections:
   Digital libraries store documents in electronic formats such as PDF, HTML, and multimedia files.
2. Automatic Indexing:
   Documents are automatically indexed using keywords, full-text indexing, and metadata, improving retrieval speed.
3. Advanced Search Capabilities:
   Users can perform keyword search, Boolean search, phrase search, and relevance-based ranking.
4. Large-Scale and Distributed Collections:
   Digital libraries handle massive and distributed collections accessible over networks like the Internet.
5. User-Centric Retrieval:
   Results are ranked based on relevance, allowing users to quickly identify useful documents.
6. Anytime, Anywhere Access:
   Information can be accessed remotely without physical constraints.

1b. Explain Software architecture of the IR System

Introduction:

The **software architecture of an Information Retrieval (IR) system** describes the structured organization of components that support document storage, indexing, querying, retrieval, and ranking of information. It defines how documents are processed offline and how user queries are handled online to retrieve relevant information efficiently

High-Level Architecture of an IR System:

The IR system architecture is broadly divided into **two major processes**:

1. **Indexing Process (Offline Process)**
2. **Searching and Retrieval Process (Online Process)**

1. Indexing Process (Offline):

The indexing process prepares documents for efficient retrieval and includes the following components:

a) Document Collection:

- Consists of raw documents such as text files, books, articles, and web pages.
- These documents form the input to the IR system.

b) Document Preprocessing:

- Documents are cleaned and normalized.
- Includes:
    - Tokenization
    - Stop-word removal
    - Stemming or lemmatization

c) Index Construction:

- Processed documents are converted into an **inverted index**.
- The inverted index maps terms to the documents in which they occur.
- This structure allows fast searching and retrieval.

d) Index Storage:

- The constructed index is stored in a repository.
- It is optimized for quick access during query processing.

2. Searching and Retrieval Process (Online):

This process handles user interaction and retrieval of results.

a) User Interface:

- Allows users to submit queries using keywords or phrases.
- Acts as the communication medium between the user and the IR system.

b) Query Processing:

- User queries undergo preprocessing similar to documents.
- Includes tokenization, stop-word removal, and stemming.
- Queries may be expanded or reformulated.

c) Query Execution:

- The processed query is matched against the inverted index.
- Matching documents are identified.

d) Ranking Module:

- Retrieved documents are ranked based on relevance.
- Ranking algorithms use factors like term frequency, inverse document frequency, and similarity measures.

e) Result Presentation:

- Ranked documents are displayed to the user.
- Includes document titles, snippets, and relevance order.

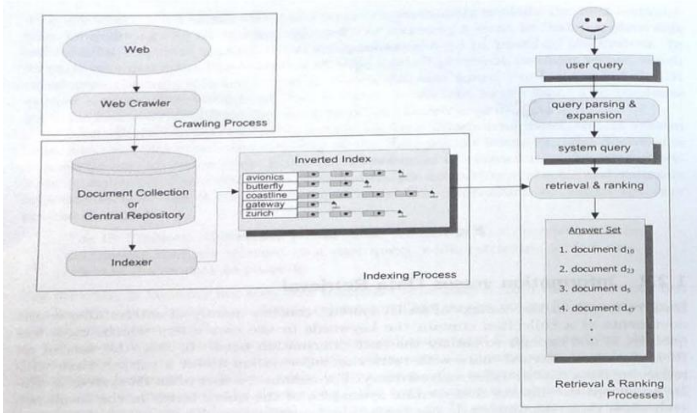
Additional Components:

Feedback Mechanism:

- Allows users to provide relevance feedback.
- Helps refine future searches and improve retrieval accuracy.

Evaluation Component:

- Measures effectiveness using metrics such as precision and recall.
- Ensures system performance and quality.



2a. Explain classic versus dynamic model of information seeking?

Information seeking refers to the process by which users search for and use information to satisfy an information need. Over time, two major models have been proposed to explain this behavior: the **Classic Model** and the **Dynamic Model** of information seeking. These models differ in how they view user needs, interaction, and the search process

1. Classic Model of Information Seeking:

The **Classic Model** views information seeking as a **linear and structured process**. It assumes that the user's information need is clearly defined from the beginning.

Characteristics:

1. The user starts with a **well-defined information need**.
2. The need is translated into a **query**.
3. The system retrieves relevant documents.
4. The user evaluates the retrieved results.
5. The process ends once the required information is found.

Key Features:

- Linear and sequential process
- Minimal user-system interaction
- Focus on system performance
- Emphasis on **precision and recall**

Limitations:

- Does not account for changing user needs
- Ignores learning during the search process
- Assumes users know exactly what they are searching for

2. Dynamic Model of Information Seeking:

The **Dynamic Model** considers information seeking as an **interactive, iterative, and evolving process**. It recognizes that users often refine their needs during the search.

Characteristics:

1. Information need may be **uncertain or vague initially**.
2. Users interact repeatedly with the system.
3. Queries are **modified and reformulated** based on results.
4. Learning occurs during the search process.
5. The process continues until the user is satisfied.

Key Features:

- Non-linear and iterative process
- High level of user-system interaction
- Supports exploration and browsing

- Emphasizes user behavior and context

Advantages:

- Reflects real-world search behavior
- Suitable for web search and digital libraries
- Supports relevance feedback and query expansion

2b. Explain design and evaluation of search interfaces

Introduction:

A **search interface** acts as the communication bridge between the user and the Information Retrieval (IR) system. Its design plays a crucial role in helping users express their information needs effectively and interpret search results efficiently. Proper evaluation of search interfaces ensures usability, effectiveness, and user satisfaction

1. Design of Search Interfaces:

The design of a search interface focuses on **simplicity, usability, and support for user search behavior**.

Key Design Principles:

1. Simple and Clear Query Input:

- Interfaces should provide a simple search box for basic users.
- Advanced options (Boolean operators, filters, fielded search) should be available for expert users.

2. Support for Query Reformulation:

- Users should be able to easily modify queries.
- Features like query suggestions, spell correction, and auto-completion improve search effectiveness.

3. Result Presentation:

- Results should be displayed in ranked order of relevance.
- Each result should include title, snippet, URL, and highlighted query terms.

4. Browsing and Navigation Support:

- Faceted navigation, filters, and categories help users explore results.
- Supports exploratory and learning-based searches.

5. Feedback Mechanisms:

- Relevance feedback allows users to mark useful results.
- Helps improve subsequent searches.

2. Evaluation of Search Interfaces:

Evaluation determines how well a search interface meets user needs and system goals.

Evaluation Methods:

1. Usability Testing:

- Observing users while they perform search tasks.
- Identifies interface problems and usability issues.

2. User-Centered Evaluation:

- Focuses on user satisfaction, ease of use, and learning effort.
- Conducted using surveys, questionnaires, and interviews.

3. Performance-Based Evaluation:

- Measures effectiveness using metrics such as:
    - Task completion time
    - Error rate
    - Success rate

4. Comparative Evaluation:

- Compares different interface designs to identify the most effective one.
- Often used during prototype testing.

Importance of Evaluation:

- Ensures the interface supports both novice and expert users
- Improves search efficiency and effectiveness
- Enhances overall user experience

Module – 2

3a. Write the characteristics of IR MODEL?

An Information Retrieval (IR) model defines the framework for representing documents, queries, and the process of matching them to retrieve relevant information. The main characteristics of an IR model are as follows:

1. **Document Representation**
    - Documents are represented in a structured form suitable for processing by the IR system.
    - Common representations include sets of terms, vectors, or probabilistic descriptions after preprocessing such as tokenization, stop-word removal, and stemming.
2. **Query Representation**
    - User information needs are expressed in the form of queries.

- Queries are represented using the same or compatible structure as documents (e.g., keyword sets or term vectors) to enable comparison.

3. **Matching Function**
   - The IR model defines a matching or similarity function to compare a query with documents.
   - This function determines how well a document satisfies the user's query.
4. **Relevance Judgment**
   - Relevance is estimated based on the matching function.
   - The model may treat relevance as binary (relevant / not relevant) or graded (degree of relevance).
5. **Ranking Capability**
   - Most IR models support ranking of retrieved documents.
   - Documents are ordered according to their relevance scores, so the most relevant documents appear first.
6. **Handling of Uncertainty**
   - IR models deal with uncertainty inherent in user queries and document content.
   - Exact matching is rare; hence partial matching and probabilistic reasoning are commonly used.
7. **Formal Framework**
   - Each IR model is based on a formal theoretical foundation such as set theory, algebra, vector space theory, or probability theory.
   - This framework defines how documents, queries, and relevance are mathematically handled.
8. **Support for Retrieval Effectiveness Evaluation**
   - IR models allow evaluation using standard measures like precision, recall, and F-measure.
   - These measures help assess how effectively the model retrieves relevant documents.

3b. EXPLAIN TF- IDF WEIGHTS

TF–IDF (Term Frequency–Inverse Document Frequency) is one of the most widely used **term weighting schemes** in Information Retrieval (IR). It reflects the importance of a term in a document with respect to the entire document collection.

1. Introduction

In Information Retrieval systems, not all terms contribute equally to the relevance of a document.

- Terms that occur frequently in a document are important.
- Terms that occur in many documents are less discriminative.

TF–IDF combines these two ideas to assign appropriate weights to terms.

2. Term Frequency (TF)

Definition:
Term Frequency measures how frequently a term appears in a document.

TF(t,d)=Number of times term t appears in document dTF(t,d) = \text{Number of times term } t \text{ appears in document } dTF(t,d)=Number of times term t appears in document d

- Higher TF indicates greater importance of the term in that document.
- Variants of TF include:
    - Raw frequency
    - Logarithmic scaling:

      TF(t,d)=1+log⁡(ft,d)TF(t,d) = 1 + \log(f_{t,d})TF(t,d)=1+log(ft,d)

    - Normalized TF (to avoid bias due to document length)

3. Inverse Document Frequency (IDF)

Definition:
IDF measures how important a term is across the entire document collection.

IDF(t)=log⁡(Ndft)IDF(t) = \log \left(\frac{N}{df_t} \right)IDF(t)=log(dftN)

Where:

- NNN = Total number of documents
- dftdf_tdft = Number of documents containing term ttt
- Rare terms get higher IDF values.
- Common terms get lower IDF values.

4. TF–IDF Weight Formula

The TF–IDF weight of a term ttt in document ddd is given by:

TF-IDF(t,d)=TF(t,d)×IDF(t)TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t)TF-IDF(t,d)=TF(t,d)×IDF(t)

This ensures that:

- A term is important if it occurs frequently in a document (**high TF**)
- A term is discriminative if it occurs in fewer documents (**high IDF**)

5. Interpretation of TF–IDF

- **High TF–IDF value:**
  The term is frequent in the document but rare in the collection → very important.
- **Low TF–IDF value:**
  The term is either rare in the document or very common across documents.

6. TF–IDF in Vector Space Model

- Documents and queries are represented as vectors of TF–IDF weights.
- Similarity between document and query is computed using measures such as **cosine similarity**.
- This helps in ranking documents based on relevance.

7. Advantages of TF–IDF

- Simple and easy to compute.
- Reduces the effect of common words.
- Improves retrieval accuracy.
- Widely used in search engines and text mining.

8. Limitations of TF–IDF

- Ignores semantic relationships between words.
- Assumes term independence.
- Does not consider word order.

9. Applications

- Search engines
- Document ranking
- Text classification
- Information filtering

4a. Explain Bayesian Network model

Bayesian Network Models are **probabilistic models** used in Information Retrieval (IR) to represent and manage uncertainty in the retrieval process. They model the relationships between documents, terms, and queries using probability theory and graphical structures.

1. Introduction

In Information Retrieval, relevance is uncertain because user queries are often incomplete or ambiguous. Bayesian Network Models address this issue by representing dependencies among variables and computing the probability that a document is relevant to a given query.

2. Definition of Bayesian Network

A **Bayesian Network (BN)** is a **directed acyclic graph (DAG)** where:

- Nodes represent random variables (terms, documents, relevance).
- Edges represent conditional dependencies between variables.
- Each node is associated with a **conditional probability table (CPT)**.

3. Basic Components of Bayesian Network Models

1. **Nodes:**
   - Query nodes represent user information needs.
   - Term nodes represent indexing terms.
   - Document nodes represent documents in the collection.
2. **Edges:**
   - Directed edges indicate dependency relationships among nodes.
3. **Conditional Probability Tables (CPTs):**
   - Quantify the strength of dependencies between connected nodes.

4. Bayesian Network Model in IR

- The model estimates the **probability of relevance** of a document given a query.
- Retrieval is based on computing:

P(D|Q)P(D \mid Q)P(D|Q)

where DDD is a document and QQQ is the query.

- Documents are ranked according to their relevance probabilities.

5. Inference Process

- User query terms are treated as observed evidence.
- Probabilistic inference is performed on the network.
- The relevance probability of each document is computed using Bayes' theorem.

6. Types of Bayesian Network IR Models

1. **Inference Network Model**
    - Separate networks for documents and queries.
    - Uses probabilistic inference to match documents to queries.
2. **Belief Network Model**
    - Represents beliefs about document relevance.
    - Updates probabilities when new evidence is available.

7. Advantages of Bayesian Network Models

- Effectively handles uncertainty in IR.
- Supports partial and probabilistic matching.
- Can incorporate term dependencies and prior knowledge.
- Provides a clear mathematical foundation.

8. Limitations of Bayesian Network Models

- High computational complexity.
- Requires estimation of many probabilities.
- Difficult to scale for very large document collections.

9. Applications

- Web search engines
- Medical and legal information retrieval
- Intelligent search systems

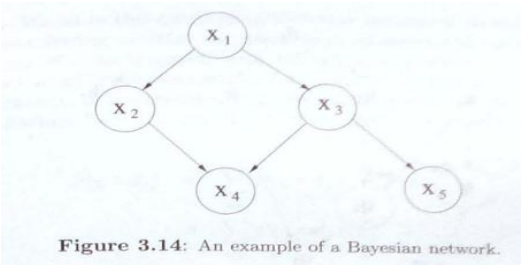


Figure 3.14: An example of a Bayesian network.

4b. Explain Extended Boolean model.

The **Extended Boolean Model** is an enhancement of the classical Boolean Information Retrieval model. It combines the **exact matching capability of the Boolean model** with the **ranking ability of vector-based models**, thereby overcoming the limitations of strict true/false retrieval.

1. Introduction

The classical Boolean model retrieves documents based on exact satisfaction of Boolean expressions (AND, OR, NOT), but it does not support partial matching or ranking. The Extended Boolean Model addresses this issue by allowing **graded relevance** and **document ranking**.

2. Limitation of Classical Boolean Model

- Retrieves documents only as relevant or non-relevant.
- No ranking of retrieved documents.
- Too strict for user queries.
- Cannot handle partial matches effectively.

3. Concept of Extended Boolean Model

- Introduced to soften Boolean operators.
- Uses **term weights** instead of binary values.
- Documents are represented as **weighted term vectors**.
- Similarity between document and query is computed numerically.

4. p-Norm Model

The Extended Boolean Model is also known as the **p-norm model**.

- The parameter ppp controls the strictness of Boolean operators.
- As p→∞p \to \inftyp→∞, the model behaves like the classical Boolean model.
- For smaller values of ppp, partial matching is allowed.

5. AND Operator in Extended Boolean Model

For an AND query with nnn terms:

SimAND(d,q)=1−(1n∑i=1n(1−wi,d)p)1/pSim_{AND}(d,q) = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - w_{i,d})^p \right)^{1/p}SimAND(d,q)=1−(n1i=1∑n(1−wi,d)p)1/p

- Penalizes documents missing query terms.
- Allows graded relevance.

6. OR Operator in Extended Boolean Model

For an OR query with nnn terms:

SimOR(d,q)=(1n∑i=1nwi,dp)1/pSim_{OR}(d,q) = \left(\frac{1}{n} \sum_{i=1}^{n} w_{i,d}^p \right)^{1/p}SimOR(d,q)=(n1i=1∑nwi,dp)1/p

- Rewards documents matching more query terms.
- Supports partial satisfaction.

7. Ranking of Documents

- Each document receives a similarity score.
- Documents are ranked in decreasing order of similarity.
- This improves retrieval effectiveness.

8. Advantages of Extended Boolean Model

- Supports partial matching.
- Allows ranking of documents.
- More flexible than classical Boolean model.
- Combines logical and vector-based approaches.

9. Limitations of Extended Boolean Model

- Selection of parameter ppp is difficult.
- More complex than classical Boolean model.
- Computationally more expensive.

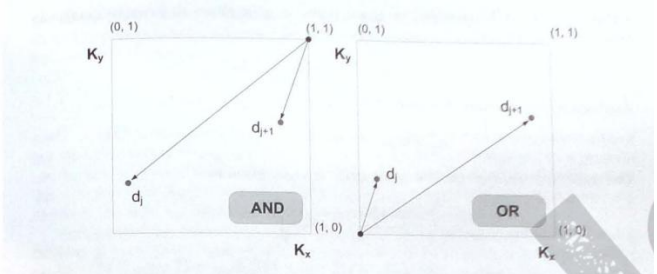


Figure 3.11: Extended Boolean logic considering the space composed of two terms k_x and k_y only.

Module -3

5a. List and explain evaluation measures in TREC Collections

Introduction:
TREC (Text REtrieval Conference) provides a standard framework for evaluating Information Retrieval (IR) systems. It uses large test collections consisting of **documents, queries, and relevance judgments**. Various **evaluation measures** are used in TREC to assess the effectiveness of IR systems in retrieving relevant documents.

Evaluation Measures in TREC

1. Precision

- **Definition:** Precision is the fraction of retrieved documents that are relevant.
- **Formula:**

  Precision=Number of relevant documents retrievedTotal number of documents retrievedPre cision = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of

documents
retrieved}}Precision=Total number of documents retrievedNumber of relevant documents re
trieved

- **Purpose:** Measures the exactness of the retrieval system.

2. Recall

- **Definition:** Recall is the fraction of relevant documents that are successfully retrieved.
- **Formula:**

  Recall=Number of relevant documents retrievedTotal number of relevant documentsRecall =
  \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant
  documents}}Recall=Total number of relevant documentsNumber of relevant documents retr
  ieved

- **Purpose:** Measures the completeness of retrieval.

3. Precision–Recall Curve

- Shows the trade-off between precision and recall at different cutoff levels.
- Used to compare performance of multiple IR systems.

4. Average Precision (AP)

- **Definition:** Average of precision values obtained after each relevant document is retrieved.
- **Importance:** Considers the ranking order of relevant documents.

5. Mean Average Precision (MAP)

- **Definition:** Mean of Average Precision over a set of queries.
- **Formula:**

  MAP=1Q∑i=1QAPiMAP = \frac{1}{Q} \sum_{i=1}^{Q} AP_iMAP=Q1i=1∑QAPi

- **Purpose:** Most commonly used TREC measure for system comparison.

6. Precision at K (P@K)

- Measures precision after retrieving top **K** documents (e.g., P@10).
- Useful for evaluating top-ranked results.

7. Mean Reciprocal Rank (MRR)

- **Definition:** Average of reciprocal ranks of the first relevant document.
- **Formula:**

  MRR=1Q∑1rank of first relevant documentMRR = \frac{1}{Q} \sum \frac{1}{\text{rank of first
  relevant document}}MRR=Q1∑rank of first relevant document1

- Used in question-answering tasks.

8. Discounted Cumulative Gain (DCG)

- Considers graded relevance and position of documents.
- Penalizes relevant documents appearing lower in the ranking.

9. Normalized DCG (NDCG)

- DCG normalized by ideal DCG.
- Enables fair comparison across queries.

10. F-Measure

- Harmonic mean of precision and recall.
- **Formula:**

  F=2PRP+RF = \frac{2PR}{P+R}F=P+R2PR

- Balances precision and recall.

5b. Explain Explicit Relevance Feedback

Introduction:
Relevance Feedback is an important technique used in Information Retrieval (IR) systems to improve search effectiveness. In Explicit Relevance Feedback**, the user explicitly indicates which retrieved documents are relevant or non-relevant, and the system uses this information to refine the query and improve subsequent retrieval results. This concept is widely discussed and evaluated in TREC-style IR systems. Explicit Relevance Feedback**

Definition:

Explicit Relevance Feedback is a process in which the user **manually marks documents** returned by the IR system as *relevant* or *non-relevant*. The system then modifies the original query using this feedback to retrieve more relevant documents.

Working of Explicit Relevance Feedback

1. The user submits an initial query to the IR system.
2. The system retrieves a ranked list of documents.
3. The user explicitly labels some documents as **relevant** and **irrelevant**.
4. The system analyzes these judgments.
5. The original query is **reformulated** based on the feedback.
6. A new set of documents is retrieved with improved relevance.

Query Reformulation (Rocchio Algorithm)

The most common method used for explicit relevance feedback is the **Rocchio Algorithm**, given by:

Qnew=αQold+β1|Dr|∑Dr−γ1|Dnr|∑DnrQ_{new} = \alpha Q_{old} + \beta \frac{1}{|D_r|} \sum D_r - \gamma \frac{1}{|D_{nr}|} \sum D_{nr}Qnew=αQold+β|Dr|1∑Dr−γ|Dnr|1∑Dnr

Where:

- QoldQ_{old}Qold = Original query vector
- DrD_rDr = Set of relevant documents
- DnrD_{nr}Dnr = Set of non-relevant documents
- α,β,γ\alpha, \beta, \gammaα,β,γ = weighting parameter

Advantages of Explicit Relevance Feedback

- Improves **retrieval accuracy**
- Helps capture **user intent**
- Enhances **precision and recall**
- Useful in **interactive search systems**

Limitations of Explicit Relevance Feedback

- Requires **extra effort from users**
- Users may provide **inconsistent judgments**
- Not suitable for users seeking quick answers
- Less practical for large-scale web search

Applications

- Digital libraries
- Research databases
- TREC evaluation experiments
- Academic and legal search systems

6a. Explain Document Pre-processing Text Operations (Transformations)

Introduction:
Document pre-processing is a crucial step in an Information Retrieval (IR) system. It involves a set of **text operations or transformations** applied to raw documents in order to convert unstructured text into a structured and searchable form. These operations help improve **retrieval efficiency, accuracy, and effectiveness** of the IR system.

Document Pre-processing Text Operations (Transformations)

1. Tokenization

- Process of breaking text into smaller units called **tokens** (words or terms).
- Removes punctuation and special characters.
- Example:
  "Information Retrieval System" → Information, Retrieval, System

2. Case Folding

- Converts all characters to **lowercase**.
- Ensures uniformity and avoids duplication.
- Example: *IR*, *Ir*, *ir* → *ir*

3. Stop Word Removal

- Removes frequently occurring words with little semantic value.
- Examples: *is, the, and, of, to*
- Reduces index size and improves performance.

4. Stemming

- Reduces words to their **root or stem form**.
- Example: *connected, connecting, connection* → *connect*
- Common algorithm: **Porter Stemmer**.

5. Lemmatization

- Converts words to their **dictionary (lemma) form**.
- Considers grammar and part of speech.
- Example: *better* → *good*

6. Normalization

- Handles variations such as:
  - Accents
  - Hyphens
  - Numbers
- Example: *e-mail* → *email*

7. Noise Removal

- Removes unwanted elements such as:
  - HTML tags
  - Scripts
  - Advertisements
- Produces clean textual content.

8. Handling Special Tokens

- Processing dates, numbers, URLs, and email IDs.
- Example: *2024* → *year* or removed depending on application.

9. Term Weighting Preparation

- Prepares terms for weighting schemes like **TF-IDF**.
- Enables ranking of documents based on importance.

10. Index Term Selection

- Selects meaningful terms for indexing.
- Eliminates redundant and irrelevant terms.

Advantages of Document Pre-processing

- Improves **search accuracy**
- Reduces **storage requirements**
- Enhances **retrieval speed**
- Produces **consistent index terms**

6b. Explain Organizing Documents

Introduction:
Organizing documents is an important function of an Information Retrieval (IR) system. It refers to the process of **structuring, classifying, indexing, and arranging documents** so that relevant information can be efficiently stored, searched, and retrieved. Proper document organization improves **search effectiveness, scalability, and user satisfaction** in both traditional and digital libraries.

Organizing Documents in Information Retrieval

1. Document Collection Formation

- Documents are first collected from various sources such as books, web pages, reports, and databases.
- Each document is assigned a **unique document identifier (DocID)**

2. Document Representation

- Documents are represented using a **set of index terms** obtained after preprocessing.
- Common representations:
    - Bag-of-Words model
    - Vector Space model
- Helps in efficient similarity computation.

3. Indexing

- Indexing is the process of creating data structures that map **terms to documents**.
- Types of indexing:
    - Inverted index
    - Forward index
- Enables fast retrieval of documents.

4. Document Classification

- Documents are grouped into **predefined categories** or classes.
- Can be:
    - Manual classification

    - Automatic classification using machine learning
- Helps in topic-based retrieval.

5. Document Clustering

- Groups similar documents together **without predefined categories**.
- Useful for browsing large document collections.
- Examples: K-means clustering, hierarchical clustering.

6. Metadata Assignment

- Metadata includes information such as:
    - Author
    - Title
    - Date
    - Keywords
- Improves document identification and filtering.

7. Ranking and Ordering

- Documents are ranked based on **relevance score**.
- Ranking models:
    - TF-IDF
    - Probabilistic models
- Ensures most relevant documents appear first.

8. Storage and Maintenance

- Organized documents are stored efficiently using databases or file systems.
- Periodic updates are required to:
    - Add new documents
    - Remove outdated documents

Advantages of Organizing Documents

- Faster information retrieval
- Reduced search space
- Improved precision and recall
- Better user navigation and browsing

Module -4

7a. Explain Simple String Horspool's Algorithm

Introduction

Horspool's algorithm is a **simple and efficient string matching algorithm** used to find the occurrence of a **pattern string (P)** in a **text string (T)**. It is an **improvement over the naive string matching algorithm** and is based on the **bad character shift heuristic**. The algorithm compares characters from **right to left** and skips sections of the text to improve performance

Basic Idea

- The pattern is aligned with the text.
- Comparison starts from the **rightmost character of the pattern**.
- If a mismatch occurs, the pattern is shifted using a **precomputed shift table**.
- The shift depends on the **rightmost occurrence of the mismatched character in the pattern**.

Shift Table Construction

Let `m` be the length of the pattern.

Rules:

1. For all characters **not in the pattern**, shift value = `m`
2. For characters present in the pattern (except last character):
   `Shift = m - 1 - position of character`
3. Last character of pattern is **not included** in table computation

This preprocessing improves search efficiency.

Algorithm Steps

1. Preprocess the pattern and build the **shift table**.
2. Align the pattern with the beginning of the text.
3. Compare pattern characters with text characters **from right to left**.
4. If all characters match → **pattern found**.
5. If mismatch occurs → shift pattern using shift table.
6. Repeat until the pattern reaches the end of the text.

Example

Text (T): `THIS IS A SIMPLE EXAMPLE`
Pattern (P): `EXAMPLE`

- Length of pattern = 7
- Shift table is constructed for characters in EXAMPL
- Algorithm skips unnecessary comparisons using shifts

Time Complexity

- **Best Case:** O(n / m)
- **Average Case:** O(n)
- **Worst Case:** O(nm) (rare)

Where:

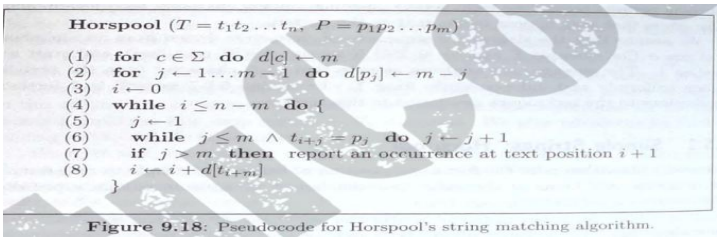
- n = length of text
- m = length of pattern

Advantages

- Simple to implement
- Faster than naive string matching
- Efficient for large texts

Limitations

- Uses only one heuristic (bad character rule)
- Not optimal for very small patterns



```
Horspool (T = t1t2...tn, P = p1p2...pm)
(1)  for c ∈ Σ do d[c] ← m
(2)  for j ← 1...m − 1 do d[pj] ← m − j
(3)  i ← 0
(4)  while i ≤ n − m do {
(5)    j ← 1
(6)    while j ≤ m ∧ ti+j = pj do j ← j + 1
(7)    if j > m then report an occurrence at text position i + 1
(8)    i ← i + d[ti+m]
       }
```

Figure 9.18: Pseudocode for Horspool's string matching algorithm.

7b. Explain Construction of Suffix Arrays for Large Texts

Introduction

A **suffix array** is a space-efficient data structure used for **full-text indexing and string matching**. It stores all suffixes of a text in **lexicographically sorted order**. For **large texts**, efficient construction algorithms are required to reduce **time and memory overhead**, especially when the text cannot fit entirely into main memory

Suffix Array – Definition

For a text T of length n, a suffix array SA is an array of integers such that:

- SA[i] gives the **starting position** of the i^{th} smallest suffix of T in lexicographical order.

Need for Efficient Construction

- Naive construction by generating and sorting all suffixes takes **$O(n^2 \log n)$** time.
- For **large texts**, this approach is inefficient.
- Hence, **linear or near-linear time algorithms** are used.

Construction Techniques for Large Texts

1. Prefix-Doubling Method

- Initially, suffixes are sorted based on their **first character**.
- In each iteration, sorting is done using **2^k-length prefixes**.
- Ranking is updated after every iteration.

- Continues until $2^k \geq n$.

Time Complexity: O(n log n)

2. Induced Sorting (Linear Time Algorithms)

- Examples: **SA-IS algorithm**
- Suffixes are classified into:
    - **L-type** (larger than next suffix)
    - **S-type** (smaller than next suffix)
- LMS (Left-most S-type) substrings are sorted first.
- Remaining suffixes are **induced** from LMS order.

Time Complexity: O(n)

3. External Memory Algorithms

Used when text size exceeds RAM.

- Text is divided into **blocks**
- Partial suffix arrays are built for each block
- Merged efficiently using disk-based techniques

Suitable for **web-scale or genomic data**

Steps in General Construction

1. Read the large text input
2. Generate suffix representations implicitly
3. Apply efficient sorting technique
4. Store only suffix **starting indices**
5. Output the final suffix array

Advantages

- Requires less space than suffix trees
- Efficient for pattern matching
- Suitable for large-scale text indexing

Applications

- Information Retrieval Systems
- DNA sequence analysis
- Text compression and plagiarism detection

8a. Explain Multiple Word Queries

Definition

A multiple word query consists of **two or more words** (terms) and retrieves documents that satisfy a specified **logical relationship** (AND / OR / NOT) between the query terms.

Example:
information retrieval
data AND mining

Types of Multiple Word Queries

1. Conjunctive Queries (AND Queries)

- Retrieves documents that contain **all query terms**
- Implemented using **intersection of posting lists**

Example:
Query: information AND retrieval
Only documents containing **both terms** are returned.

2. Disjunctive Queries (OR Queries)

- Retrieves documents that contain **at least one query term**
- Implemented using **union of posting lists**

Example:
Query: data OR mining

3. Boolean Queries

- Combine AND, OR, and NOT operators
- Uses Boolean logic for retrieval

Example:
information AND retrieval NOT database

Processing Multiple Word Queries

1. Each query term is looked up in the **inverted index**
2. Corresponding **posting lists** are fetched
3. Posting lists are merged using:
    - **Intersection** for AND queries
    - **Union** for OR queries
4. Documents satisfying query conditions are returned

Efficient merging improves search performance.

Optimizations

- **Ordering posting lists by increasing length** to reduce comparisons
- **Skip pointers** to speed up list intersection
- **Early termination** when matches are exhausted

Advantages

- Improves retrieval accuracy

- Reduces irrelevant documents
- Supports complex user information needs

Limitations

- More computation compared to single-word queries
- Performance depends on posting list size

Applications

- Web search engines
- Digital libraries
- Enterprise search systems

8b. Explain Structure of Tries and Suffix Trees.

Introduction

Tries and **Suffix Trees** are important **tree-based data structures** used in **Information Retrieval and string processing**. They support efficient **searching, indexing, and pattern matching** operations, especially for textual data.

1. Structure of Tries

Definition

A **Trie** (also called a prefix tree) is an **ordered tree structure** used to store a set of strings, where:

- Each node represents a **character**
- Paths from the root represent **prefixes of words**

Structural Characteristics

- The **root node** represents an empty string
- Each edge is labeled with a **character**
- Each node may have multiple children
- A **terminal marker** indicates the end of a word

Example

Words: to, tea, ten

- Common prefixes share the same path
- Saves space by avoiding repetition

Operations Supported

- Insertion
- Exact word search
- Prefix search

Advantages

- Fast lookup: **O(m)** time, where m is word length
- Efficient prefix-based searching

Limitations

- High memory usage due to many nodes
- Inefficient for very large alphabets

2. Structure of Suffix Trees

Definition

A **Suffix Tree** is a **compressed trie** that stores **all suffixes of a given text**. Each suffix is represented as a path from the root to a leaf.

Structural Characteristics

- Root represents empty string
- Each leaf corresponds to a **suffix starting position**
- Edges are labeled with **substrings**, not single characters
- Internal nodes represent **common substrings**

Example

Text: banana$
Suffixes:

- banana$
- anana$
- nana$
- ana$
- na$
- a$
- $

All suffixes are stored in a single tree.

Key Properties

- Contains **n leaves** for a text of length n
- Depth is proportional to text length
- Compact representation compared to trie

Advantages

- Pattern matching in **O(m)** time
- Efficient for substring search
- Useful for longest repeated substring problems

Limitations

- Complex to construct
- High memory overhead

Module -5

9a. Explain Cluster-Based Architecture

Definition:

Cluster-based architecture is an architecture where the document collection is divided into several **clusters**, and queries are processed by searching only the most relevant clusters instead of the entire collection.

Need for Cluster-Based Architecture:

- Large web collections cannot be handled efficiently by a single machine.
- Reduces **search space** and **response time**.
- Enables **parallel processing** of queries.
- Supports **scalability** by adding more clusters.

Architecture Components:

1. **Query Interface:**
   Accepts the user query and forwards it to the broker.
2. **Broker (Coordinator):**
   - Determines which clusters are relevant for the given query.
   - Distributes the query to selected clusters.
   - Collects and merges results from clusters.
3. **Clusters:**
   - Each cluster contains a subset of the document collection.
   - Documents may be grouped based on topic, content similarity, or random partitioning.
4. **Local Search Engines:**
   - Each cluster has its own local index and search engine.
   - Executes the query on local documents.
5. **Result Merger:**
   Combines ranked results from all clusters into a final ranked list.

Working of Cluster-Based Architecture:

1. User submits a query through the query interface.
2. The broker analyzes the query and selects relevant clusters.
3. The query is forwarded to selected clusters.
4. Each cluster processes the query independently using its local index.
5. Results are returned to the broker.
6. The broker merges and ranks the results.
7. Final ranked results are presented to the user.

Advantages:

- Faster query processing due to reduced search space.
- High scalability and flexibility.
- Efficient use of distributed resources.
- Suitable for large-scale web search engines.

Disadvantages:

- Overhead in cluster selection.
- Possible loss of relevant documents if clusters are poorly formed.
- Increased system complexity.

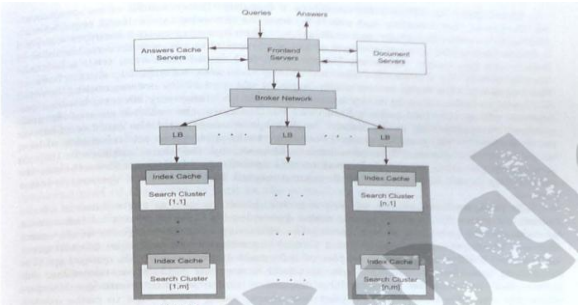


Figure 11.7: Cluster-based architecture for the search module based on [484]. Each cluster contains an index of the whole collection, i.e., the index is partitioned among the m servers in the cluster. The n clusters are used to produce n replicas of the whole index.

9b. Explain Retrieval Task

Definition:

A retrieval task is the activity of selecting and ranking documents from a document collection based on their relevance to a given user query.

Key Components of a Retrieval Task:

1. **Document Collection:**
   A large set of documents such as web pages, articles, reports, or databases from which information is retrieved.
2. **User Query:**
   A formal representation of the user's information need, usually expressed using keywords or phrases.
3. **Indexing Mechanism:**
   Documents are preprocessed and indexed to enable fast and efficient retrieval.
4. **Retrieval Model:**
   A mathematical or probabilistic model (e.g., Boolean, Vector Space, Probabilistic model) used to match queries with documents.
5. **Ranking Function:**
   Assigns a relevance score to documents and orders them accordingly.

Types of Retrieval Tasks:

1. **Ad-hoc Retrieval:**
   - User submits a one-time query.
   - System retrieves relevant documents from a static collection.
   - Common in web search engines.

2. **Known-Item Retrieval:**
    - User searches for a specific document already known to exist.
    - Example: searching for a particular research paper.
3. **Filtering / Routing:**
    - Documents are continuously matched against long-term user profiles.
    - Used in news filtering and email spam detection.
4. **Interactive Retrieval:**
    - User refines the query based on results returned by the system.
    - Involves relevance feedback.

Steps Involved in a Retrieval Task:

1. User formulates a query.
2. Query is processed (tokenization, stop-word removal, stemming).
3. Indexed documents are matched against the query.
4. Relevance scores are computed.
5. Documents are ranked.
6. Top-ranked documents are presented to the user.

Objectives of Retrieval Task:

- Retrieve maximum relevant documents (**high recall**).
- Retrieve minimum non-relevant documents (**high precision**).
- Provide ranked results based on relevance.
- Reduce response time and improve user satisfaction.

Challenges in Retrieval Task:

- Ambiguous and short queries.
- Large-scale document collections.
- Vocabulary mismatch between query and documents.

10a. Explain Static vs Dynamic Architecture

Introduction:

Search engine architectures can be broadly classified into **static architecture** and **dynamic architecture** based on how documents are indexed, updated, and retrieved. These architectures differ in terms of **index maintenance, update handling, and query processing**.

1. Static Architecture

Definition:

Static architecture is a search engine architecture in which the **document collection and index remain unchanged** for a long period of time. Updates are performed infrequently and usually offline.

Characteristics:

- Index is built once and rarely updated.
- Document collection is assumed to be static.

- Queries are processed only on the existing index.
- Suitable for small or stable collections.

Working:

1. Documents are collected.
2. Index is constructed offline.
3. Queries are processed using the static index.
4. Results are retrieved without considering frequent updates.

Advantages:

- Simple design and implementation.
- Faster query processing due to stable index.
- Low maintenance overhead.

Disadvantages:

- Cannot handle frequent document updates.
- Results may become outdated.
- Not suitable for web-scale systems.

Examples:

- Digital libraries
- Archived document collections
- Offline CD/DVD search systems

---

2. Dynamic Architecture

Definition:

Dynamic architecture is a search engine architecture in which the **document collection and index are continuously updated** to reflect additions, deletions, and modifications of documents.

Characteristics:

- Supports frequent updates.
- Index is incrementally modified.
- Handles dynamic content like web pages.
- Requires complex index management.

Working:

1. New documents are continuously crawled.
2. Index is updated incrementally.
3. Queries are processed on the latest index.
4. Results reflect recent changes.

Advantages:

- Provides up-to-date search results.
- Suitable for large and evolving collections.
- Essential for web search engines.

Disadvantages:

- Complex architecture.
- Higher computational and storage cost.
- Index updates may affect query performance.

Examples:

- Google Search
- Bing
- News search engines

10b. Explain Single Hierarchy vs Multiple Hierarchy

Introduction:

In Information Retrieval and search engine architectures, document organization can be done using **hierarchical structures**. Based on how documents are classified and stored, hierarchies are categorized into **single hierarchy** and **multiple hierarchy** systems. These approaches differ in flexibility, scalability, and retrieval efficiency.

1. Single Hierarchy

Definition:

Single hierarchy is an organization method where each document belongs to **only one category or path** in the hierarchical structure.

Characteristics:

- Tree-like structure.
- Each document has a single parent.
- Clear and simple classification.
- No overlapping categories.

Working:

- Documents are classified into a fixed hierarchy.
- Retrieval follows one predefined path.
- Queries are matched within the assigned category.

Advantages:

- Simple to design and maintain.
- Fast retrieval due to limited search paths.
- Easy navigation.

Disadvantages:

- Rigid structure.
- A document relevant to multiple topics must be placed in only one category.
- Poor handling of multi-topic documents.

Examples:

- Traditional library classification systems.
- File directory structures.
- Static subject trees.

2. Multiple Hierarchy

Definition:

Multiple hierarchy is an organization method where a document can belong to **multiple categories or paths** simultaneously.

Characteristics:

- Graph-like structure.
- Documents may have multiple parents.
- Supports overlapping and cross-domain topics.
- More flexible classification.

Working:

- Documents are indexed under multiple hierarchies.
- Queries can traverse multiple paths.
- Improves retrieval of multi-topic documents.

Advantages:

- Higher recall due to multiple classifications.
- Flexible and user-friendly.
- Suitable for complex and dynamic document collections.

Disadvantages:

- Increased storage and indexing cost.
- Complex maintenance.
- Possible redundancy.

Examples:

- Web directories.

- Tag-based systems.
- Modern digital libraries.