

ISR Endsem

Unit 3

Performance Evaluation

In case of performance evaluation of data retrieval system, shorter the response time and smaller the space used, better the system is. Thus, it is tradeoff between time and space.

In case of performance evaluation of Information retrieval system, relevance of retrieved document is important, besides time and space.

In IR, since user query is inherently vague, the retrieved document are not exact answers and have to be ranked according to the relevance to the query.

Need of performance evaluation measures in IR system

Performance evaluation measures are essential in an **Information Retrieval (IR) system** to:

- Determine how well the system retrieves relevant documents while minimizing irrelevant ones.
- Compare different IR systems or algorithms to choose the best performing one.
- Identify areas of improvement, such as relevance ranking or query formulation, to enhance user satisfaction.
- Ensure retrieved results meet the user's expectations in terms of relevance and quality.
- Provide metrics that help in system tuning and maintenance over time.

Various performance Evaluation measures are: -

The standard approach to information retrieval system evaluation revolves around the notion of relevant and non-relevant documents.

In response to a user's query, an IR system searches its document collection and returns an ordered list of responses. It is called the retrieved set or ranked list.

The system employs the search strategy or algorithm and measures the quality of a ranked list.

A better search strategy yields a better ranked list, and better ranked lists help the user fill their information need.

Precision and **Recall** are the basic measures used in evaluating search strategies.

Precision is defined as the ratio of number of relevant documents retrieved to the number of total retrieved documents from the query.

Recall is defined as the ratio of number of relevant documents retrieved to the number of relevant documents in the database.

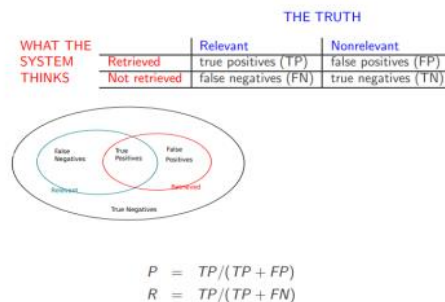
Precision is the fraction of retrieved documents that are relevant, whereas Recall is the fraction of relevant documents that are retrieved.

- Precision (P) is the fraction of retrieved documents that are relevant

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

- Recall (R) is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$



Trade-off between Recall and Precision

Recall focuses on retrieving all relevant documents, even at the cost of including some irrelevant ones.

Thus, high recall may result in retrieving many irrelevant documents, reducing precision.

Precision focuses on retrieving only relevant documents, even if some relevant ones are missed.

Thus, high precision may exclude some relevant documents, reducing recall.

High Recall, Low Precision: - Useful in scenarios like legal document discovery or medical research, where missing a relevant document is costly.

High Precision, Low Recall: - Useful when the user needs only the most relevant documents, such as top search engine results.

To balance the trade-off, composite metrics like **F-Score** or **NDCG** are used to evaluate the system holistically.

Mean Reciprocal Rank (MRR)

MRR evaluates the quality of ranked search results by measuring how early the first relevant document appears in the ranking.

It provides an average reciprocal rank across multiple queries.

Higher MRR indicates that relevant documents appear earlier in the results.

MRR measures the effectiveness of ranking in a search result.

Formula:

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{\text{Rank}_i}$$

- N : Total number of queries.
- Rank_i : Rank position of the first relevant document for the i^{th} query.

F-Score

F-Score combines Precision and Recall into a single measure to balance their trade-off.

Thus, it provides harmonic mean of the precision and recall, favoring systems that balance both.

A high F-Score indicates good overall performance in retrieving relevant documents.

Formula:

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

- Precision: Proportion of retrieved documents that are relevant.
- Recall: Proportion of relevant documents that are retrieved.

Normalized Discounted Cumulative Gain (NDCG)

NDCG is performance evaluation metric used in IR system to measure the quality of ranked search results considering both the relevance and position of documents.

It rewards systems that rank highly relevant documents higher in the results.

Key Concepts in NDCG are: -

- Relevance score: Each document is assigned a relevance score based on how relevant it is to the query.

- Discount factor: Relevance score is discounted logarithmically as the rank increases to give higher importance to top-ranked documents.
- Cumulative Gain: - Sum of relevance scores in the retrieved results.

$$CG = \sum_{i=1}^n rel_i$$

- Discounted Cumulative Gain (DCG): DCG applies the discount factor to relevance scores based on the rank.

$$DCG = rel_1 + \sum_{i=2}^n \frac{rel_i}{\log_2(i+1)}$$

- Ideal DCG (IDCG): DCG calculated for the ideal ranking where all relevant documents are ranked in descending order of relevance.
- Normalized DCG (NDCG): NDCG is obtained by normalizing the DCG by the IDCG. It ranges between 0 and 1, with 1 indicating perfect ranking.

$$NDCG = \frac{DCG}{IDCG}$$

NDCG considers both the relevance of documents and their ranks, making it a more robust measure for evaluating ranked retrieval systems.

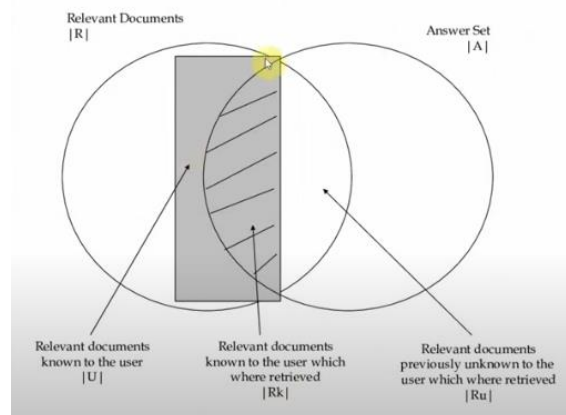
A high NDCG value indicates that relevant documents are retrieved and ranked effectively.

User Oriented Measures

Recall and Precision assumes that the set of relevant documents for a query is independent of the users. However, different users might have different relevance interpretations.

To cope up with this problem, user-oriented measures have been proposed.

Let's consider a reference collection,



There are 4 different types of user-oriented measures: -

- Coverage ratio

Coverage ratio can be defined as the fraction of the documents known to the user to be relevant which actually have been retrieved.

A high coverage ratio indicates that the system is finding most of the relevant documents that the user expected to see.

$$\text{Coverage Ratio} = \frac{|R_k|}{|U|}$$

- Novelty ratio

Novelty ratio can be defined as the fraction of the relevant documents retrieved which was unknown to the user.

A high novelty ratio indicates that the system is finding many new documents which were previously unknown to the user.

$$\text{Novelty Ratio} = \frac{|R_u|}{|R_u| + |R_k|}$$

- Relative Recall

Relative Recall is the ratio between the number of relevant documents found and the number of relevant documents the user expected to find.

When user finds as many documents as he expected, then he will stop search, and relative value will be 1.

$$\text{Relative Recall} = \frac{|R_k| + |R_u|}{|U|}$$

- Recall Effort

Recall effort is the ratio of the number of relevant documents the user expected to find, and the number of documents examined in an attempt to find the expected relevant documents.

$$\text{Recall Effort} = \frac{|U|}{|A|}$$

Visualization in Information System

Information Visualization is the process of representing data in a visual and meaningful way so that a user can better understand it.

Information Visualization is an art and therefore relies on the following aspects of design: -

- The Subject Matter: Information or data being represented.
- The Story: The concept being portrayed in the visualization.
- The Goal: Meeting the purpose with the right visualization.
- The Visual: Using key elements of structure and design.

Starting Points

Search interfaces must provide users with good ways to get started.

Usually, users do not begin by creating a long, detailed expression of their information needs.

Types of Starting points: -

- Lists: Often used sources are stored on a favorites list also known as a bookmark list or hotlist on the web.
- Overviews: It helps users get started, directing them into general neighborhoods, after which they can navigate using more detailed descriptions.
- Examples: Another way to help users get started is to start them off with an example of interaction with the system. This technique is also known as retrieval by reformulation.
- Automated source selection: Automatically selecting the best source for a query is to automatically send a query to multiple sources and then combine the result from the various systems in some way.

Query Specification

A query in Information Retrieval is a formal statement or request made by a user to retrieve specific information or documents from a database or collection.

It can be a keyword, phrase, or structured statement designed to match relevant data.

To execute a query, users must select collections, metadata descriptions or information sets against which the query is to be matched and must specify words or phrases that can be compared against the information in the collection.

The system creates a set of documents, metadata or other information type that match the query specification in some sense and displays the results to the user in some form.

Shneiderman identifies five types of primary human computer interaction styles: /
Various techniques used to specify query in information visualization:

- Command line interface: It provides a means of expressing instructions to the computer directly, using function keys, single characters, abbreviation or whole-word commands.
- Menus: The set of available options are displayed on the screen and selected by the mouse or numeric or alphabetic keys. Menus can be nested hierarchically.
- Natural language: Natural language is very difficult for a machine to understand. It is ambiguous, syntactically and semantically.
- Question/Answer, query dialogue: It is a simple mechanism for providing input to an application in a specific domain rather than generating an answer from scratch, question answering systems attempts to link a natural language query to the most pertinent sentence, paragraph, or page of information that has already been written.
- Form fills and spreadsheets: Used primarily for data entry but also useful in data retrieval. The display resembles a paper form, with slots to fill in.

Document Context

Document Surrogates: A document surrogates is a limited representation of full documents. Some systems provide users with a choice between a short and detailed view.

Query terms hits within Document Content:

Systems in which the user can view the full text of a retrieved document, it is often useful to highlight the occurrences of the terms or descriptors that match those of the user's query.

Different data visualization interfaces are:

- KWIC (keyword-in-context): The KWIC feature allows user to search for any number of terms relevant to the keywords and view them in a tabular overview along with the words that appear before and after their respective context.
- TileBars: TileBar is a data visualization interface used in conjunction with a search feature, query terms and expository text.
- SeeSoft: The SeeSoft visualization represents text in a manner resembling columns of newspaper text, with one line text on each horizontal line of the strip. The representation is compact.

Query term hits between Documents:

Rather than showing how query terms appear within individual documents as is done in KWIC interfaces and TileBars, these systems display an overview or summary of the retrieved documents according to which subset of query the term they contain.

Some visualization tools used are:

- InfoCrystal: InfoCrystal can be used as a visualization tool as well as a visual query language to help users search for information. It shows how many documents contain each subset of query terms.
- VIBE and Lyberworld: Graphical presentations that operates on similar principles are VIBE and Lyberworld. Query terms are placed in an abstract graphical space.
- SuperBook: The SuperBook system makes use of the structure of a large document to display query term hits in the context.

User Relevance Judgement

User Relevance Judgement is the process of evaluating whether a retrieved document or piece of information satisfies the user's information need for a given query.

This assessment can be binary (relevant or not) or graded (e.g., highly relevant, moderately relevant, or irrelevant).

Measuring relevance of documents with respect to a user's query is at the heart of Information Retrieval (IR), where the user's relevance judgement criteria have been recognized as multi-dimensional.

This measure is used to measure the performance of any retrieval system and visualize results of retrieval system.

It is measured using following factors: -

- Time spent with the document.
- Clicked on link or not.
- Explicit survey of relevance.
- Feedback of relevance from user.

There can be implicit or explicit user relevance judgement done:

Explicit Relevance: Here user is asked specifically about the relevance of the retrieved results.

Implicit Relevance: Here without informing the user, some behavioral aspects of the users are collected. These aspects can be time spent on link or clicking of a link, etc and on basis of its relevance is decided.

Group Relevance Judgments

- Group Relevance Judgments refer to the evaluation of information relevance performed collaboratively by a group of users, experts, or a community rather than an individual.
- To improve objectivity and account for diverse perspectives, especially in cases where relevance is subjective or context dependent.
- For example, in academic databases, a panel of experts may collaboratively judge the relevance of research articles for a query.

Pseudo-Relevance Feedback

Pseudo-Relevance feedback is an automated method used to enhance the performance of an information Retrieval system.

It assumes that the top-ranked documents retrieved for an initial query are relevant and uses them to refine the query for better results.

Steps in Pseudo-Relevance Feedback

- Initial Retrieval: The IR system retrieves a set of documents for the original query.
- Term Extraction: Relevant terms are extracted from the top-ranked documents.
- Query Expansion: The original query is expanded or reformulated using the extracted terms.
- Re-retrieval: The refined query is used to retrieve a new set of documents, ideally with improved relevance.

It is widely used in search engines to improve the quality of results automatically based on the top-ranked documents.

Interface Support for Search Process

Search process gives results of document which are relevant to the user query. These results need to be presented properly to the user.

Results can be sorted by relevance or by considering frequently used documents.

Interface should allow tracking of implicit as well as explicit user relevance parameters.

The user interface designer must make decisions about how to arrange various kinds of information on the computer screen and how to structure the possible sequences of interactions.

It refers to the design and functionality provided by the user interface to an IR system to assist users in effectively formulating queries, exploring results, and refining searches.

In context of Information visualization, it emphasizes the use of visual tools and techniques to enhance user interaction and understanding of the search process.

Example systems are: -

The InfoGrid Layout: The Information Grid is a framework for building information access applications that provides a user interface design and an interaction model. It is an typical example of a monolithic layout for an information access interface.

The SuperBook Layout: The layout of InfoGrid is quite similar to that of SuperBook. The main difference is that SuperBook retains the table of contents-like display in the main left-hand pane, along with indicators of how many documents containing search hits occur in each level of the outline. Like InfoGrid the main pane of the right-hand side is used to display selected documents.

The DLITE Interface: The DLITE system makes a number of interesting design choices. It splits functionality into two parts: Control of search process and display of results. The control portion is graphical direct manipulation display with animation.

Unit 4

Distributed IR

Distributed Information Retrieval refers to the process of retrieving information from multiple, geographically dispersed collections or databases.

Instead of relying on a centralized repository, the retrieval system operates over several distributed sources, each containing its own subset of data.

The distributed computing system can be viewed as a MIMD parallel processor with relatively slow inter-processor communication channel and the freedom to employ a heterogeneous collection of processors in the system.

Distributed system typically consists of a set of processes, each running on a separate processing node.

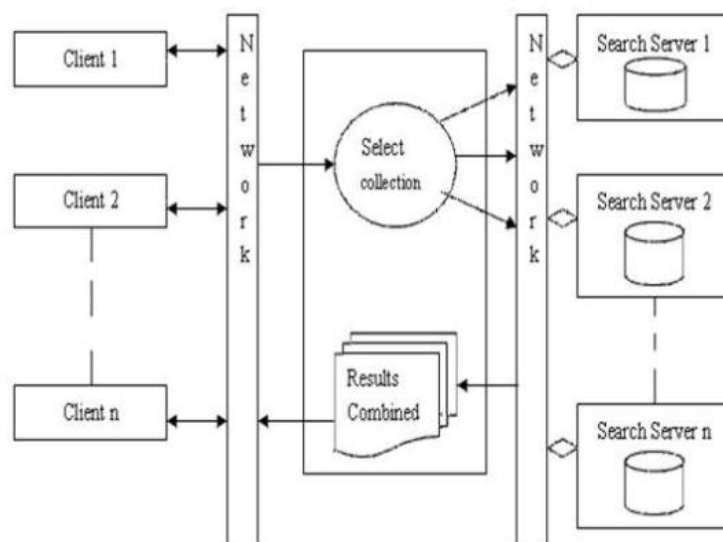
A broker process is responsible for: -

- Accepting client requests.
- Distributing the requests to the servers.
- Collecting intermediate results from the servers.
- Combining the intermediate results into a final result for the client.

Distributed computing uses multiple computers connected by a network to solve a single problem.

A distributed computing system can employ a heterogeneous collection of processors in the system.

In fact, a single processing node in the distributed system could be a parallel computer in its own right.



Architecture of Distributed Information Retrieval

Automatic Feature Extraction in Distributed IR

It is a process of identifying and extracting meaningful features from the data present in the distributed sources, without manual intervention.

These features are used to index documents, process queries and rank results effectively.

Steps in Automatic Feature Extraction are: -

- **Data Preprocessing:** Cleans and standardizes the data for feature extraction. It includes techniques like stop-word removal, stemming, etc.
- **Feature Identification:** Automatically identifies key features from the various data collections, such as text features, numerical features, Image features, etc.
- **Feature Selection:** Determines which features are most relevant to queries. Techniques like TF-IDF, PCA, etc are used.
- **Feature Representation:** Converts features into numerical formats that can be processed by retrieval algorithms. Techniques like Vector Space model are used for representing text.

Collection Partitioning in Distributed IR

It refers to dividing a large corpus or dataset into smaller subsets, known as collections or partitions, which are distributed across multiple storage locations or servers.

Each partition can be searched independently, contributing to the efficiency and scalability of the overall system.

In Distributed IR, the volume of data can be immense, making it impractical to store and manage everything in a single location, thus Collection Partitioning is required.

Collection Partitioning allows:

- Handling large-scale datasets by distributing the workload.
- Reducing search latency by focusing queries on specific partitions.
- Enabling simultaneous searches across different collections.
- Balancing the computational and storage load across servers.

Steps in Collection Partitioning:

- **Analyzing Data:** Identifying the key features for partitioning (e.g., topics, geography, time).
- **Partitioning Strategy:** Choosing an appropriate partitioning method based on data and use case.
- **Indexing Partitions:** Creating separate indexes for each collection to facilitate efficient searches.

- Query Routing: Designing mechanisms to route queries to the most relevant partitions.

Collection Partitioning can be done on various basis such as: -

- Topic-based Partitioning: Data is partitioned based on topics or subject areas.
- Geographical Partitioning: Data is partitioned based on geographic regions.
- Chronological Partitioning: Data is divided based on time or date ranges.
- Format-based Partitioning: Data is divided based on its format or type.
- Random Partitioning: Data is randomly distributed across partitions without any specific criteria.

Advantages of Collection Partitioning: -

- Reduces the search space for individual queries.
- Allows distributed storage and processing of large datasets.
- Enables simultaneous searches across partitions.
- Isolates failures to specific partitions, ensuring system reliability.

Challenges in Collection Partitioning: -

- Queries that span multiple partitions may require additional processing.
- Accurately identifying relevant partitions for each query is complex.
- Ensuring even distribution of data and search workloads.
- Handling real-time additions and modifications to the collections.

Example of Collection Partitioning: -

Scenario:

A library system has millions of books divided into collections:

- **Topic-based:** Fiction, Science, History.
- **Chronological:** Books published in the 20th century vs. 21st century.
- **Geographical:** Local and international collections.

When a user queries "modern science books," the system may direct the query to:

- The "Science" topic-based collection.
- The 21st-century partition under chronological division.

Results from both collections are then merged and presented.

Collection partitioning is a foundational concept in Distributed IR that ensures efficiency, scalability, and fault tolerance in handling vast and diverse datasets.

By dividing data into manageable partitions and intelligently routing queries, it optimizes both resource utilization and user satisfaction.

Source Selection

Source Selection is the process of determining which of the distributed document collections are most likely to contain relevant documents for the current query and therefore should receive the query for processing.

Since searching all distributed sources can be inefficient, source selection helps in narrowing down the search to the most relevant sources, improving retrieval efficiency.

Steps in Source Selection are: -

- Query Analysis: Understand the user query and its intent.
- Source Ranking: Rank sources based on their likelihood of containing relevant information for the query, using precomputed statistics like, term frequency in collections, relevance score, etc.
- Source Selection: Choose a subset of sources with the highest relevance scores, by avoiding the irrelevant sources.
- Result Aggregation: Gather and combine the results from the selected sources. Apply ranking mechanism to present the most relevant results.

Techniques used in Source Selection are: -

- Query-based sampling: Collects samples of documents from each source using queries, then estimates the relevance of each source based on these samples.
- Statistical Approaches: Uses metrics like term distribution or document frequency across sources to rank collections.
- Learning-based methods: Employs machine learning models to predict the relevance of a source for a given query.
- Federated Search: Combines results from multiple sources into a single ranked list.

Example: -

Scenario: A user searches for "machine learning algorithms" in a distributed library system containing the following collections:

- Collection 1: Textbooks on computer science.
- Collection 2: Research papers in engineering.
- Collection 3: Articles on history and literature.

Process:

- Query Analysis: The system identifies "machine learning algorithms" as relevant to computer science and engineering.
- Source Ranking:

1. Collection 1 has high relevance due to overlapping terms like "machine learning."
 2. Collection 2 is also ranked high due to technical content.
 3. Collection 3 is deemed irrelevant.
- Source Selection: The system chooses Collections 1 and 2 for the search.
 - Result Aggregation: Results from both collections are merged and ranked before being presented.

Query Processing

Query processing refers to the steps involved in interpreting and executing a user's query to retrieve relevant information from a database or information retrieval system.

It includes analyzing the query, matching it against the stored data, and returning the results in a ranked or structured manner.

In the context of Distributed IR, query processing is more complex because it involves multiple independent data sources distributed across different locations or systems.

Steps in Query Processing are: -

- Query Analysis: The system interprets the query to understand the user intent, identifies key terms, entities and other semantic features.
- Source Selection: Determines which source are most relevant for the query, by ranking sources based on their likelihood of containing relevant information.
- Query Translation: Adapts the query to match the structure and format of each selected source. Accounts for differences in indexing, data schemas, or search algorithms across sources.
- Distributed Search Execution: Sends the adapted query to the selected sources. Searches are performed in parallel across the distributed collections.
- Result Collection: Gathers the results from different sources and normalizes it.
- Result Aggregation: Merges results from various sources into a single ranked list. Combines scores using techniques like rank aggregation or voting-based methods. Then presents the results to the user in an understandable and actionable format.

Techniques for optimizing Query Processing in Distributed IR: -

- Federated Search: Combines results from distributed sources into a unified list.
- Caching: Stores frequently queried results to reduce redundant processing.
- Statistical Source Ranking: Uses pre-computed metadata to rank sources for faster source selection.

- **Parallel Processing:** Leverages distributed computing to handle searches simultaneously across sources.
- **Result Normalization:** Ensures consistency in ranking scores from different sources.

Multimedia IR

Multimedia IR is a branch of Information Retrieval that focuses on searching and retrieving multimedia content such as images, videos, audio, and graphics instead of text-only documents.

Multimedia IR systems use various techniques to index, analyze, and retrieve content based on features such as color, texture, shape, and sound.

It deals with multi-modal data like videos, audios, and images and analyzes multimedia content using features like color histograms, shapes, and radio frequency.

It often requires rich interfaces for specifying queries and visualizing results.

Multimedia data is more complex and computationally intensive to process than textual data.

A multimedia system is characterized by the computer-controlled generation, manipulation, presentation, storage, and communication of independent discrete and continuous media.

Multimedia data is large and affects the storage, retrieval and transmission of multimedia data. For these reasons, the development of multimedia system is considerably more complex than a traditional information system.

Multimedia IR systems require some form of database schema because several multimedia applications need to structure their data at least partially.

The architecture of Multimedia IR system consists of the following components:

- **Data Acquisition:** Collects multimedia content from various sources, such as image databases, video repositories and audio archives. Ensures metadata is also captured alongside raw multimedia content.
- **Feature Extraction:** Extracts features from multimedia content, which are then used for indexing and retrieval. Converts multimedia data into numerical vectors for efficient processing.
- **Data Modelling:** Builds a structured representation of multimedia data. Creates models to describe the relationship between multimedia objects and their features.

- Indexing: Indexes extracted features for fast retrieval, by using common techniques like inverted files, spatial access methods, etc.
- Query Processing: Supports different query types, like text-based queries, content-based queries, etc. Processes and translates queries into feature-based representations.
- Similarity Matching: Matches user queries against indexed multimedia features. Uses distance metrics to measure the similarity.
- Result Ranking: Combines multiple factors such as feature similarity, metadata, and user feedback and rank retrieved results based on their relevance to the query.

Data Modelling

A data model should be defined by which the user can specify the data to be stored into the system.

A multimedia IR system should be able to:

- Represent and store multimedia objects in a way that ensures fast retrieval.
- Deal with different kinds of media.
- Deal with semi-structured data.
- Extract features from multimedia objects.

It also provides a model for the internal representation of multimedia data.

Traditional DBMSs are targeted to support conventional data.

Whereas the multimedia data are not encoded into attributes provided by the data schemas. It requires large storage, and the content is difficult to analyze and compare.

Addressing data model in Multimedia IR systems consists of two main tasks:

- A data model should be defined by which user can specify the data to be stored into the system.
- The system should provide a model for internal representation of multimedia data.

The performance OODBMS in terms of storage techniques, query processing, and transaction management is not comparable to that of relational DBMS.

Multimedia Data Support in Commercial DBMS

To represent multimedia data, current relational DBMS support variable length data types.

Data type supported by commercial DBMS is mostly non-standard and each DBMS vendor uses the different names for such data types and provides support for different operation on them.

Oracle DBMS provides the VARCHAR2 data type to represent variable length character strings. The maximum length of VARCHAR2 data is 4000 bytes. The RAW and LONG Raw data types are used for data that is not to be interpreted by ORACLE.

Sybase SQL server supports IMAGE and TEXT data types to store images and unstructured text respectively and provides a limited set of functions for their searching and manipulation.

Most commercial relational DBMSs vendors are investing a lot of effort in extending the relational model with the capability of modelling complex objects, typically for the object-oriented context. SQL3 is an example of this type.

In SQL3, each type of specification consists of both attributes and function specification. User defined functions can be either visible from any object or only visible in the object they refer to.

Both single and multiple inheritances can be defined among user defined types and dynamic late binding is provided.

SQL3 also provides three types of collection of data types: sets, multisets and lists.

SQL3 provides a restricted form of object identifier that supports sharing and avoids data duplication. But SQL3 has not yet been officially published most commercial products.

Content based queries on text documents can be combined with traditional queries in the same SQL statement and can be efficiently executed due to the use of indexing techniques specific for texts.

Illustra provides 3D and 2D spatial data blades for modelling spatial data. The supported data types include boxes, vectors, quadrangles, etc. and supported operations are INTERSECT, CONTAINS, OVERLAPS, CENTER and so on.

Spatial data blades also implements R-trees for performing efficient spatial queries. The text data blade provides data types for representing unstructured text and performing content based queries.

The object relational technology and its extensive type system is now starting to be widely used both in industrial and research projects.

The MULTOS Data Model

MULTOS stands for Multimedia Office Server.

MULTOS is a multimedia document server with advanced document retrieval capabilities, developed in the context of an ESPRINT project in the area of office systems.

It is based on client server architecture.

Three types of document servers are supported:

- Current servers
- Dynamic servers
- Archive servers

All these servers differ in storage capacity and document retrieval speed.

These servers support filling and retrieval of multimedia objects based on document collections, document types, document attributes, document text, and images.

The MULTOS data model allows:

- The representation of high-level concepts presents in the documents contained in the database.
- The grouping of documents into classes of documents having similar content and structure.
- The expression of conditions on free text.

Each document is described by:

- Logical structure: Determines arrangements of logical document components (titles, introduction, chapter, section, etc).
- Layout structure: A deal with the layout of the document content and its components (pages, frames, etc).
- Conceptual structure: Allows a sematic oriented description of the document content.

In MULTOS the representation of documents and operations on them, are based on a formal model.

At the beginning a standardized document presentation was assumed, i.e. the ODA model.

The Office Document Architecture (ODA) is a standard defined by ISO. ODA gives the formal description of the document composition (logical structure) and formal device independent description of the document presentation (layout structure).

The logical structure associates the content of the document with a hierarchy of logical objects, whereas the layout structure associates the same content with a hierarchy of layout objects.

Documents having similar conceptual structures are grouped into conceptual types. Conceptual types are maintained in a hierarchy of generalization. The types can be strong (completely specifies the structure of its instance) or weak (partially specifies the structure of its instance).

Component of unspecified types are called spring component types.

MULTOS also provides a sophisticated approach to deal with image data. Initially an image analysis process is performed. It consists of two phases:

- Low level image analysis.
- High level image analysis.

During low level image analysis phase, the basic objects composing a given image and their relative positions are identified.

The high level image analysis phase deals with image interpretation according to the Dempster-Shafer Theory of evidence.

In the last phase of image analysis, images are described in terms of the objects recognized, with associated belief and plausibility values, and the classes to which they belong.

Query Languages

Queries in relational or object-oriented database systems are based on an exact match mechanism, by which the system is able to return exactly those tuples satisfying some well specified criteria given in the query expression.

When a query is submitted, the features of the query object are matched with respect to the features of the objects stored in the database and only the objects that are more similar to the query are returned to the user.

Query languages in multimedia IR are tools and methodologies that allow users to interact with a multimedia database or retrieval system to specify their search criteria for multimedia objects such as text, images, audio, and video.

These languages are designed to handle the complexity of multimedia data, supporting both traditional text-based queries and content-based multimedia queries.

It supports for multiple data types, thus handles queries involving text, images, audio, video, or a combination of these.

It allows user to search based on content of multimedia objects rather than just metadata.

It enables complex queries involving logical, spatial, temporal, and semantic relationships between multimedia objects.

It also allows user to specify queries in a natural language-like format or through graphical user interfaces.

In designing a multimedia query language, following points are considered:

- How the user enters his/her request to the system, i.e. which interface is provided to the user for query information.
- Which conditions on multimedia objects can be specified in the user request.
- How uncertainty, proximity and weights impact the design of the query language.

Query languages supporting retrieval of multimedia objects are:

- SQL3 Query language: -
 - SQL3 is an extension of SQL, includes advanced data types to handle multimedia objects such as images, audio, and video.
 - It allows structured queries to retrieve multimedia content stored as BLOBs (Binary Large Objects) or custom multimedia data types.
- MULTOS Query language: -
 - MULTOS provides robust support for querying multimedia objects based on their content. It enables the combination of content-based and metadata-based search criteria.
 - It allows users to search images by color similarity or videos by motion patterns rather than relying solely on textual metadata.

Thus, query languages in multimedia IR are critical for enabling users to retrieve multimedia content effectively.

Background – Spatial Access Methods

Spatial Access Methods are indexing techniques used in Multimedia IR systems to store and efficiently retrieve spatial data, such as images, videos and geographical information.

These methods are useful for handling data that has inherent spatial characteristics, such as coordinates, shapes, or regions.

Data that represents objects in a spatial domain is called spatial data. For e.g. Points, Lines, Polygons, etc.

The purpose of SAMs is to organize spatial data in a way that supports efficient querying and retrieval.

Some of the Spatial Access Method Techniques are: -

- R-Tree: It is a tree-based data structure for indexing multi-dimensional data. Used to group nearby objects and minimize the number of disk accesses during a query.
- Quad-Trees: It divides a 2D space into quadrants recursively. It is suitable for applications like image indexing where spatial granularity is critical.
- K-D Trees: It is a binary tree for indexing points in a k-dimensional space. Used for nearest-neighbor queries or range searches.
- Grid-based indexing: The spatial region is divided into uniform grids. Each cell holds data corresponding to objects within its boundaries.

Thus, Spatial Access Methods form a backbone of indexing and querying in multimedia IR systems, enabling efficient retrieval of spatial-oriented data.

Generic Multimedia Indexing Approach (GEMINI)

GEMINI is a framework designed to support the content-based retrieval of multimedia objects by indexing and comparing their features efficiently.

This approach is widely used for managing and querying high-dimensional data, such as images, audio, or video, in a scalable and effective way.

We should design fast search algorithms that locate objects that match a query object, exactly or approximately.

An obvious solution is to apply sequential scanning for each and every object by computing its distance. However, this can be slow and time-consuming.

GEMINI aims to provide faster alternative and is based on two ideas:

- A quick-and-dirty test, to discard quickly the vast majority of non-qualifying objects.
- The use of spatial access methods to achieve faster-than-sequential searching.

The reason for the distance computation being expensive is that multimedia objects can have a very large dimensionality.

The quick-and-dirty test is designed to reduce this dimensionality to more manageable proportions, often to only one or two dimensions.

Effectively, each multimedia object is projected onto a lower dimensional space by extracting some important features.

The distance between the query and collection objects is measured in this lower-dimension space, with little computation effort.

Any object that is distant from the query object by more than a decided threshold is disqualified from further consideration.

Finally, each collection object that was not disqualified is then fully compared to the query object in the original high dimensional space.

Spatial access methods involve segmenting the multimedia objects into smaller, logically-cohesive components, and storing these in a data structure such that the original object can be easily reconstructed.

For example, a video clip could be divided into scenes, while a still image could be stored as a fixed number of overlapping rectangular segments.

When evaluating a query, objects can be compared on a component-by-component basis, possibly in short-circuit.

GEMINI Algorithm:

- Determine the distance function $D()$ between two objects.
- Find one or more numerical feature-extraction functions, to provide a 'quick-and-dirty' test.
- Prove that the distance in the feature space lower-bounds the actual distance $D()$, to guarantee correctness.
- Use a SAM to store and retrieve the f-D feature vectors.

One-Dimensional Time Series

A time series is a sequence of data points recorded over time at uniform intervals.

In One-dimensional time series data is represented as a single variable evolving over time, such as stock prices, temperature readings, or audio signals.

The time series is represented as a series of values such as, $T = \{t_1, t_2, \dots, t_n\}$, where t_i is the value of time i . For e.g. ECG signals, weather patterns.

Applications in Multimedia IR: -

- Audio Retrieval: Based on features like pitch, amplitude, and frequency.
- Motion Patterns in Videos: Analyzing temporal variations in object movement.

It uses various metrics for similarity matching such as Euclidean distance, Distance Time Wrapping, etc.

Various feature extraction techniques like Fourier Transform, Wavelet Transform, etc are used for compressing and analyzing the signals.

Indexing methods such as Symbolic Aggregate Approximation and dimensionality reduction are used for efficient querying of time-series databases.

Two-Dimensional Color Images

A 2D color image is represented as a grid of pixels, where each pixel contains color information typically in RGB (Red, Green, Blue) or other color spaces (e.g., HSV, YUV).

Pixels are represented as a 2D arrays: $I[x][y]$, where x, y are spatial coordinates, and pixel values denote intensity/color.

Various features such as color, texture, shape, etc are extracted from the image.

Techniques like Content-Based Image Retrieval (CBIR) are used to retrieve images similar to an input image.

Spatial access methods such as R-trees are used for storing the image features.

Similarity measures like cosine similarity, histogram intersection, etc are used for searching the images.

Trends and Research Issues

Emerging Trends:

- AI and Machine Learning: Using neural networks for better feature extraction and retrieval.
- Semantic Search: Bridging the gap between low-level features and high-level user queries.
- Personalization: Tailoring search results based on user behavior and preferences.

Challenges in Multimedia IR:

- Scalability: Efficiently handling large-scale multimedia datasets.
- Semantic Gap: Mapping low-level features (e.g., color) to high-level user concepts (e.g., "sunset").
- Real-Time Retrieval: Meeting the demands for fast and accurate results in dynamic systems.

Research Directions:

- Developing cross-modal retrieval systems (e.g., searching for an image using text).
- Enhancing feature extraction using unsupervised learning techniques.
- Exploring context-aware retrieval that incorporates location, time, and user preferences.

Unit 5

Web Searching

Introduction

The World Wide Web was developed by Tim Berners-Lee in 1990 at CERN to organize research documents available on the Internet.

It combined the idea of documents available by FTP with the idea of hypertext to link documents.

Web documents are known as pages, each of which can be addressed by an identifier called a uniform resource locator. Web pages are usually grouped into sites, i.e. sets of pages published together.

Web searching refers to the process of retrieving relevant information from the vast and continuously growing collections of web pages available on the internet.

This is primarily facilitated by search engines that help users locate specific content by entering query.

Clients use browser application to send URIs via HTTP to servers requesting a Web page.

Web pages are constructed using HTML or other markup language and consist of text, graphics, sounds plus embedded files.

Server responds with the requested web page. Client's browser renders the web page returned by the server.

The entire system runs over standard networking protocols such as TCP/IP, DNS, etc.

By combining the robust algorithms, efficient architecture, and user-friendly interfaces, web searching has become a cornerstone of modern information retrieval.

Web Challenges in IR

- Handling billions of web pages requires extensive computational and storage resources.
- Many dynamically generated sites are not indexable by search engines, this phenomenon is known as the invisible web.
- The ordering of results is not always solely by relevance, but sometimes influenced by monetary contributions. It is difficult for business model.
- Some sites use tricks to manipulate the search engine to improve their ranking for certain keywords, this is known as search engine spamming.

- Frequent updates and new page additions make maintaining up-to-date indices challenging.
- Diverse content types such as text, images, videos, etc complicate uniform indexing and retrieval.
- Semi-structured or unstructured data demands advanced extraction techniques.
- Detecting and filtering low-quality or manipulative content is essential.
- Analyzing links is resource intensive and may miss semantic connections.
- Balancing tailored results with user data privacy requires careful strategies.
- Crawling, indexing, and real-time querying impose significant network and processing demands.

Web Characteristics

In characterizing the structure and content of the Web, it is necessary to establish precise semantics for Web concepts.

The web is vast, dynamic, and complex information space. To understand and effectively utilize it, two main aspects are considered:

- Measuring the Web
- Modelling the Web

Measuring the Web: -

Internet and particular web are dynamic in nature, so it is difficult task to measure.

Web explosion is due in no small part to the extended application of an axiom known as Moore's Law.

In the mid 1980s, you might spend all afternoon visiting your friends before dropping by the bank and grocery, and then go out to dinner and a show after.

Today, your banking, shopping and chat with your friends are all readily handled from your tablet or phone, and all of it can be accomplished in less time than it takes to say, "The Internet made me hermit".

Of course, back then, it was something of a privilege to be online, but the net gained traction really quickly.

While just 16 million people were using web in 1995, this figure had leapt to 50 million three year later, one billion by 2009 and more than twice that last year.

In 1993 there was an estimated 130 website online, which jumped to 10 thousand by 1996. Fast forward to 2012, and there are 634 million websites competing for your attention, and it's still growing.

As web pages are constructed using HTML or other markup languages, if we assume that the average HTML pages have 5 KB and that there are 300 million web pages, we have at least 1.5 terabytes of text.

Modelling the Web: -

Can we model the document characteristics of the whole web? Answer is “yes”.

The Heap's and Zipf's law are also valid in the web.

Normally the vocabulary grows faster, and the word distribution should be more biased. But there are no such experiments on large Web collections to measure these parameters.

Heaps's law: An empirical rule which describes the vocabulary growth as a function of the text size. It establishes that a text of n words has a vocabulary of size $O(n^B)$ for $0 < B < 1$.

Zipf's law: An empirical rule that describes the frequency of the text words. It states that the i th most frequent word appears as many times as the most frequent one divided by i^o , for some $o > 1$.

Search Engines

A search engine is a program designed to help in finding information stored on a computer system such as World Wide Web or a personal computer.

The search engine allows one to ask for content meeting specific criteria and retrieves a list of references that match those criteria.

Search engines are among the most important applications or services on the web, and they are becoming a primary entry point for discovering web pages.

The architectures of search engines are of following types: -

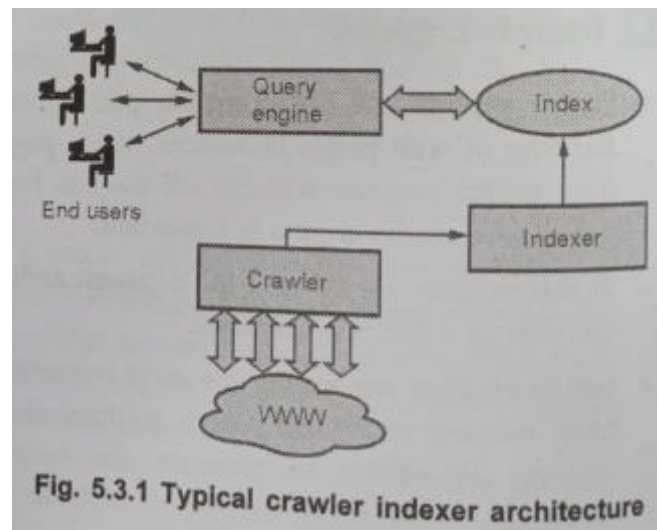
Centralized Architecture

Centralized crawler indexer architecture is used by most of the search engines.

Using crawlers, information is gathered into a single site, where it is indexed, the site then processes all user queries.

This system has its own data collection mechanism, and all the data are stored and indexed in a conventional database system.

Although many web search engines download web pages and provide service by thousands of servers, they all belong to this kind according to their basic architecture.



Centralized architecture consists of following components: -

- Crawlers: Crawlers gathers data from web pages by navigating through hyperlinks. It automatically downloads the web pages and extracts their content and continuously updates the database to reflect changes on the web.
- Index: Stores the information gathered by the crawlers in a structured format, thus enables efficient and fast retrieval of relevant documents. It uses data structures like inverted indices to map keywords to documents.
- Query Engine: It processes user queries by analyzing and interpreting them. Matches the queries with indexed data to retrieve relevant results. It also implements ranking algorithms to prioritize the most relevant documents.
- User Interface: It is the front-end system through which users interact with the search engine. It allows users to input queries and view results in a user-friendly format.

Problems using this architecture are: -

- Difficult to gather the data because of dynamic nature of the Web.
- The high load on Web servers.
- Large volume of data, thus it could be difficult for crawler to cope with Web growth.
- Communication link problem.

Distributed Architecture

When the data source is large enough that even the metadata can't be efficiently managed in a database system, then we can choose distributed system.

Distributed information retrieval system does not have its own actual record database. It just indexes the interface of sub database system.

When receiving the query from a user, main system will instantly obtain the records from sub databases through their search interfaces.

The limitation of this system is that the number of sub databases can't be many, otherwise the search speed can't be ensured.

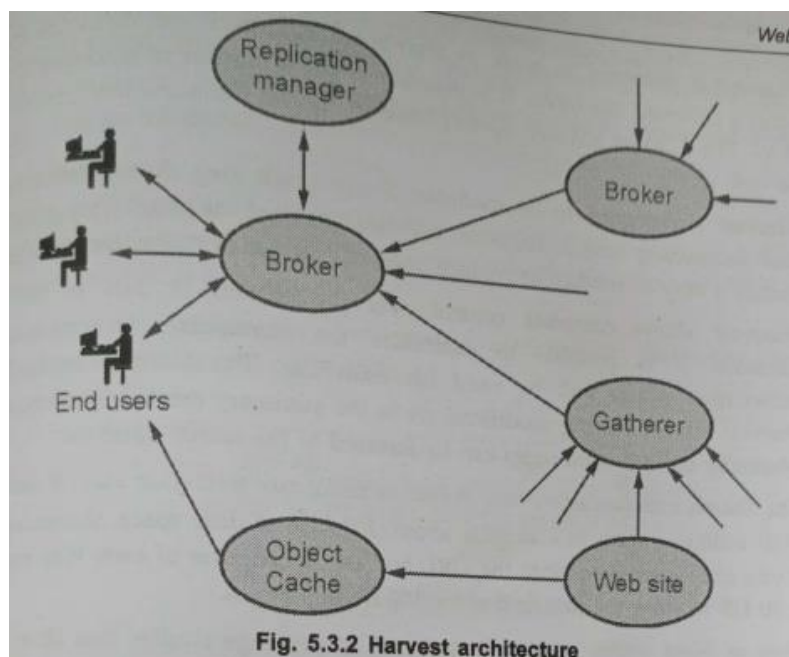
Harvest is an example of distributed architecture.

Harvest is a distributed crawler-indexer architecture which addresses the main problems in crawling and indexing the Web:

- Web servers get requests from different crawlers of search engines which increase the server's load.
- Most of the entire objects retrieved by the crawlers are useless and discarded.
- No coordination exists among the crawlers.

Harvest is designed to be a distributed search system where multiple machines work together to handle the load which single machine could not handle.


Harvest also can be used to save bandwidth by deploying gatherers near the data source and exchanging the summarized data which usually is much smaller than original data.



Components of the architecture are: -

- **Gatherers:** It extracts information from the documents stored on one or more Web servers. It can handle documents in many formats like HTML, PDF, Postscript, etc.
- **Broker:** It provides indexing mechanism and query interface.
- **Replicator:** It is used to replicate servers, thus enhances user base scalability.
- **Object Cache:** It reduces network and server load. It also reduces response latency when accessing Web pages.

Difference between Centralized and Distributed architecture of search engine: -

Aspect	Centralized Architecture	Distributed Architecture
Definition	All data and processes are managed in a single system.	Data and processes are spread across multiple servers.
Data Storage	Stored in a single location.	Stored across distributed servers with partitioning.
Scalability	Limited scalability; struggles with large-scale data.	Highly scalable; easily handles large-scale data.
Fault Tolerance	Low fault tolerance; a single point of failure affects the system.	High fault tolerance due to replication and redundancy.
Speed and Efficiency	Slower for large data as all processes are centralized.	Faster as tasks are parallelized across multiple systems.
Implementation	Easier to implement and maintain.	Complex implementation and maintenance.
Cost	Lower initial setup cost.	Higher setup and operational costs.
Communication Overhead	Minimal since all operations occur in one location.	Higher due to data transfer and synchronization between servers.
Use Case	Suitable for small-scale or moderately sized search systems.	Suitable for large-scale search systems like Google.
Examples	Early search engines with simpler architectures. 	Modern large-scale search engines (e.g., Google, Bing).

User Interfaces

In search engine, user interfaces are crucial for enabling effective interaction between users and the system.

Search engines have a user interface for expressing the query and for displaying the answers.

The interface is tightly coupled with the retrieval software, and it varies from system to system.

Two important aspects of the user interfaces of search engine are query interface and answer interface.

Query interface: -

Basic query interface is a box where one or more words can be typed.

Although user would expect that a given sequence of words represents the same in all search engine, but it does not.

Google, the most popular search engine, essentially uses the same user-interface it started out with in 1997.

In the technology world where things get outmoded by the day, this is an amazing record for longevity.

This record is more surprising as the classic web-search interface is not exactly a model of usable design.

Another problem is that the logical view of the text is not known, i.e. some search engines use stopwords, some do stemming and some are not case sensitive.

Answer interface: -

Answer interface usually consists of a list of the ten top ranked web pages.

Each entry in list includes some information about the document it represents.

Typical information includes the URL, size, the date when the page was indexed, and the couple of lines with its contents.

Some search engines allow the user to change the number of pages returned in the list and the amount of information per page, but in most cases, this is fixed or limited to a few choices.

Most of the search engines also have an option to find documents similar to each web page in the answer.

Ranking

The primary challenge of a search engine is to return results that match a user's needs. A word will potentially map to millions of documents, but how to order them is the main problem.

Most of the search engines use variations of the Boolean or vector model to do ranking.

Each search engine uses a proprietary 'ranking algorithm' that attempts to instantly build a list of highly appropriate responses to your query.

Since each search engine applies its own formula to a unique database of information, results and relevancy rankings will always vary from search engine to search engine.

Yuwono and Lee propose three ranking algorithms in addition to the classical scheme, they are Boolean spread, Vector spread, and Most cited.

Boolean spread and vector spread are the normal ranking algorithms of the Boolean and vector model extended to include pages pointed to by a page in the answer.

The last most cited is based only on the terms included in pages having a link to the pages in the answer.

Hyperlink information is also used by some of the new ranking algorithms. The number of hyperlinks that point to a page provides measures of its popularity and quality.

Page ranking is a method used by search engines to determine the relevance and importance of web pages in response to a user's query.

It assigns a score to each page, which helps rank the pages in the search results.

Role of Page ranking in Web searching: -

- Ensures that the most relevant and credible pages appear higher in the search results.
- Helps users quickly find information by displaying the most useful pages at the top.
- Ranks millions of pages efficiently, providing meaningful results from an overwhelming amount of data.
- Websites strive to produce high-quality and user-friendly content to improve their ranking.
- Factors like keyword relevance, backlink quality, user behavior, and freshness of content contribute to ranking.
- High-ranking pages often attract more traffic, making them prime spots for advertisements.
- It combines relevance to user queries with the popularity of pages and adjusts ranking based on user preferences, location and browsing history.

Crawling the Web

A web crawler is also known as web spider.

Crawler is a program which browses the World Wide Web in a methodical, automated manner.

Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches.

It starts with a list of URLs to visit, as it visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, recursively browsing the Web according to a set of policies.

Web crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code.

Also, crawlers can be used to gather specific type of information from Web pages, such as harvesting e-mail addresses.

Role of crawler in Web searching: -

- **Discovering Web Content:** Automatically explores and fetches web pages by following links.
- **Building an index:** Collects and stores data about web pages to make them searchable.
- **Keeping Data fresh:** Regularly revisits web pages to ensure the index is up to date.
- **Handling large scale Web Content:** Efficiently processes millions of pages in a scalable manner.

Strategies used by Web Crawlers: -

- **Breadth-First Search:** Crawls pages layer by layer, starting from a seed page and visiting linked pages sequentially.
- **Depth-First Search:** Explores a path deeply by following links on a single page before backtracking to other links.
- **Priority-based Crawling:** Assigns priorities to pages based on criteria like page rank, update frequency, or URL structure.
- **Focused Crawling:** Targets specific topics or domains by analyzing content relevance during crawling.
- **Incremental Crawling:** Regularly revisits pages to detect updates while ignoring unchanged content.

Components of a Web Crawler: -

- **Seed URLs:** A list of starting points for the crawler, typically well-known or popular pages.
- **Downloader:** Fetches web pages from the internet using HTTP/HTTPS protocols.
- **Scheduler:** Manages the queue of URLs to be visited, prioritizing based on crawling strategies.

- URL Frontier: A data structure that stores URLs to be crawled and ensures no duplicate visits.
- Parser: Used to extract links and metadata from the crawled web pages.
- Content Storage: Stores the fetched web pages or extracted content for indexing.
- Indexer Interface: Communicates with the indexing module to update the search engine's database.
- Politeness module: Ensures adherence to website-specific crawling rules, like robots.txt.

Indices in Web Searching

Indices or Indexes are critical structures in search engines, designed to store and retrieve data efficiently from large datasets.

They help improving the speed and relevance of web searching.

Indexing is the process of organizing web data to enable quick and accurate retrieval by the search engine.

The purpose of indices: -

- To improve query processing speed.
- To reduce the time complexity of searching large datasets.
- To support ranked retrieval and ensure relevance.

Types of indices: -

- Forward indexing: Stores information about the words found on each document. For e.g. Document-to-term mapping.
- Inverted Indexing: - It is commonly used in web searching. Stores a mapping of terms to the documents where they occur. For e.g. Term-to-document mapping.

The structure of an inverted index includes: -

- Term Dictionary: A list of all unique words (vocabulary) in the dataset.
- Posting List: A list of documents where each term occurs, along with metadata such as frequency, position, etc.

Steps in Building an Index are: -

- Tokenization: Breaking the text into individual words or terms.
- Normalization: Converting the terms to a standard format (e.g., lowercase).
- Stopword Removal: Removing common words (e.g., the, is).
- Stemming: Reducing terms to their base or root form.
- Index construction: Creating the inverted index using the processed terms.

Index is complemented with a short description of each web page. It includes title, creation date and size, first line, etc.

There are many different parts to a search engine index, such as design factors and data structures.

The design factors of a search engine index, designs or outline the architecture of the index and decide how the index actually works.

The different data structures like a suffix tree, inverted index, citation index, etc are used to create an index.

These different types come together with architecture and build a search engine index that is ready to use and quickly returns the results that the visitor is looking for.

Web directories

A web directory is a curated collection of websites, organized into hierarchical categories and subcategories, allowing users to browse and locate relevant websites based on topics.

Unlike search engines which rely on automated crawling, web directories are typically managed by human editors.

The purpose is to provide organized and categorized listing of websites for easy navigation and discovery.

Web directories are structured hierarchically, starting from broad categories for e.g. Science, Technology, etc and narrowing down to specific subcategories, for e.g. Artificial Intelligence, Web development, etc.

Websites are indexed based on topics rather than keywords. Human reviewed entries ensure quality relevance.

Some of the examples of Web directories are:

- DMOZ (Open Directory Project): A once-popular directory edited by volunteers.
- Yahoo Directory: One of the earliest web directories, now discontinued.

Advantages: -

- High quality and spam free listings.
- Useful for exploring websites on a specific topic without needing precise search queries.

Disadvantages: -

- Limited coverage compared to search engines.
- Time-intensive to maintain and update entries.

Although less common today, web directories were once essential tools for discovering websites in the early days of the internet.

Meta-searchers

It is a web server that sends a given query to several search engines and web directories and collects the answers and unifies them.

Thus, it is a type of search engine that aggregates search results from multiple other search engines and provides a unified results list.

It does not maintain its own database of web pages but relies on other search engines for data.

Working of Meta-Searchers: -

- User enters a query into the meta-search engine.
- The query is simultaneously sent to multiple search engines.
- Results from these engines are collected and merged by the meta-search engine.
- The aggregated results are ranked and presented to the user in a single interface.

Metasearch engines present the results of their searches in one of two ways:

Single list: Most of metasearcher display multiple-engine search results in a single merged list, from which duplicate entries have been removed.

Multiple lists: Some metasearchers do not collate multiple-engine search results but display them instead in separate lists as they are received from each engine. Duplicate entries may appear.

Example of meta searchers are Dogpile, Metacrawler, Yippy, Savvysearch, etc.

Thus, Meta-searchers offer a unique approach to web searching by combining the strengths of multiple search engines.

Searching using Hyperlinks

A hyperlink is a reference or navigating element in a document that connects one resource to another.

Hyperlinks are essential for web navigation, enabling users to jump from one web page to another or from one location within a document to another.

Structure of Hyper link: -

A hyperlink typically consists of two parts:

- **Anchor Text:** The clickable text displayed to the user.
- **URL:** The actual link destination, which could be a web page, file, or a location within a document.

For e.g. `Visit Example`, here Visit Example is an Anchor text and href is a URL/Target.

Hyperlinks are utilized in web search engines to discover, index and rank web pages.

The relationship between web pages, created by hyperlinks, forms the backbone of the web searching.

Role of hyperlinks in Web Searching: -

- Search engines crawlers follow hyperlinks to discover new web pages and update existing ones in the index.
- The crawler starts from a list of known URLs and follows links to gather information about additional pages.
- Hyperlinks provide context and authority to web pages.
- Techniques like PageRank uses hyperlink structure to determine a page's importance.
- Anchor text associated with the hyperlinks gives clues about the content of the linked page, aiding in relevance assessment.

Hyperlinks are fundamental to the functioning of the web and web search engines.

By enabling navigation and providing relational data, they serve as a critical component for crawling, indexing and ranking in web searching.

Web Scraping

Web scraping is a process of extracting data from websites.

It involves fetching a web pages content and parsing it to retrieve specific information.

It is commonly used for data analysis, competitive research, and creating datasets from publicly available information.

Steps involved in Web scrapping are: -

- Send a request: Use HTTP libraries to request the webpage's content.
- Parse the Response: Extract relevant data from the HTML structure using tools like BeautifulSoup or XPath.
- Store the Data: Save the extracted data into structured format like CSV, JSON, or a database for further use.

Applications of Web Scraping are:

- Market Research: Extracting product prices, reviews, and ratings from e-commerce sites.
- News Aggregation: Collecting headlines and articles from multiple sources.
- Real Estate Analysis: Gathering property listings from websites.
- Social Media Monitoring: Analyzing posts, hashtags, or trends from platforms.

Web searching example using Python: -

The example below scrapes the titles of articles from a news website,

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
# Step 1: Send a request to the website
url = "https://example-news-site.com"
response = requests.get(url)

# Step 2: Parse the webpage content
soup = BeautifulSoup(response.text, "html.parser")

# Step 3: Extract specific data (e.g., article titles)
titles = soup.find_all("h2", class_="article-title")

for title in titles:
    print(title.text)
```

Some challenges in Web Scraping are: -

1. Many websites prohibit scraping in their terms of service.
2. Sites may block or limit access to prevent scraping.
3. Ensure data is used responsibly and within legal bounds.

Thus, web scraping is a powerful tool for extracting valuable information from websites.

Request Module

The request module is a Python library used to send HTTP requests and interact with web servers.

It is a powerful and easy-to-use library for retrieving web pages and APIs, making it a fundamental tool for web scraping and data extraction tasks.

It has simple syntax and provides an intuitive API for making HTTP requests such as GET, POST, PUT, DELETE, etc.

It allows customization of requests using headers and query parameters.

It easily accesses the response content, status codes, and metadata.

It also maintains session cookies for seamless interaction.

Some common Methods in Requests: -

- `requests.get(url)`: Retrieves the content of the webpage.
- `requests.post(url, data)`: Sends data to the server (e.g., form submissions).
- `requests.head(url)`: Retrieves metadata without the body content.
- `requests.put(url)`: Updates resources on the server.

Beautiful Soup Library

The Beautiful Soup Library is a python tool for parsing and extracting data from HTML and XML documents.

It provides simple methods for navigating, searching and modifying the document tree.

It can also works with a poorly formatted HTML documents.

It allows traversal of HTML document tree to locate desired elements.

It can extract tags, attributes and text efficiently.

It is often used in conjunction with requests module to fetch and parse web pages.

Some commonly used methods are: -

- `find(tag, attributes)`: Finds the first matching tag.
- `find_all(tag, attributes)`: Finds all matching tags.
- `select(css_selector)`: Finds elements using CSS selectors.
- `get_text()`: Extracts text content from elements.

Thus, the requests module fetches web content efficiently, while the Beautiful Soup library enables parsing and extracting data from that content. Together, they form a powerful toolkit for web scraping and data extraction tasks in Python.

UNIT 6

Advanced Information Retrieval

Basic XML Concepts

XML stands for eXtensible Markup Language.

It is emerging as a standard for exchanging data on the Web. It enables separation of content (XML) and presentation (XSL).

The XML standard was created by W3C to provide an easy to use and standardized way to store self-describing data.

It is widely used for representing structured and semi-structured data in a format that is both human-readable and machine-readable.

XML uses tags to describe the structure and meaning of the data.

XML is a cross-platform format that can be used on any system.

Users can define their own tags, making XML flexible for various applications.

It organizes data in a tree-like structure with elements and sub-elements.

As it separates data and presentation, it focuses on the content rather than how it should be displayed.

Structure of XML Document: -

An XML document consists of:

- Prolog: Contains metadata such as XML version and encoding type.
- Root Element: The top-level element that encapsulates all other elements.
- Child Elements: Sub-elements within the root element, representing data.
- Attributes: Provides additional information about an element.
- Text Content: The actual data or value inside an element.

Applications of XML: -

1. Web Services: Data exchange in SOAP and REST APIs.
2. Configuration Files: Used for storing configuration settings.
3. Document Storage: Represents documents in publishing and e-commerce.
4. Database Markup: Stores semi-structured data in XML-enabled databases.

Challenges in XML Retrieval

XML retrieval focuses on retrieving specific parts of XML documents, rather than treating them as a whole.

This is essential for databases, web applications and other systems that store information in XML format.

Various challenges in XML Retrieval are: -

- XML documents often have different structures, schemas, and tags for similar information, making uniform retrieval difficult.
- Determining the correct level of detail (e.g., retrieving a paragraph vs. an entire section) is complex.
- Traditional indexing techniques are inadequate for XML since the structure (hierarchy) must also be indexed alongside content.
- Queries need to consider both content and structure (e.g., "Find books where the author is 'John' within <library>"). Processing such complex queries is computationally expensive.
- Determining the relevance of nested elements (e.g., <section> vs. <paragraph>) adds complexity to ranking.
- Changes in document schema (e.g., adding new tags or attributes) can disrupt retrieval processes.
- XML documents can be very large, requiring efficient storage and retrieval mechanisms.
- Developing suitable metrics to evaluate XML retrieval systems is challenging because the structure and content relevance must both be measured.
- Multiple ways to represent the same data (e.g., attributes vs. nested elements) create inconsistencies in querying.
- XML often needs to work alongside other formats like JSON or relational databases, complicating retrieval.

Vector Space Model for XML Retrieval

The Vector Space Model is widely used model in information retrieval that represents documents and queries as vectors in a multidimensional space.

For XML retrieval, this model is adapted to account for the hierarchical structure and granularity of XML documents.

The dimensions of vector space in unstructured retrieval are vocabulary terms, whereas they are lexicalized subtrees in XML retrieval.

There is tradeoff between the dimensionality of the space and the accuracy of query results.

Instead of treating an XML document as a single entity, individual elements or nodes are considered as separate units. Each unit is represented as a vector of terms.

Each term in the vector is assigned a weight using TF-IDF to reflect its importance.

Here term frequency (TF) refers to importance of the term within the specific XML document, whereas Inverse Document Frequency (IDF) indicates the rarity of the term across all XML elements.

The similarity between the query vector and XML element vectors is calculated using measures such as cosine similarity.

The model considers the structural relationships between elements, such as the parent-child or sibling relationships.

Structural information may be integrated as additional features in the vector or through structural constraints during retrieval.

Steps in VSM for XML retrieval: -

- Parse the XML document into its hierarchical components, and extract text content from individual elements for vector representation.
- Build an index of terms for each XML element, incorporating their structural hierarchy.
- Represent the query as a vector, queries may include both content-based and structure-based components.
- Calculate similarity scores between the query vector and the vectors of relevant XML elements.
- Rank the XML elements or nodes based on their similarity scores and return the most relevant elements to the user.

Advantages: -

- Support partial matches and ranking of results.
- Targets specific elements rather than entire documents.
- Accounts for both the textual content and structural hierarchy of XML.

By adapting VSM for XML, retrieval systems can effectively handle structured and semi-structured data, catering to the hierarchical nature of XML documents.

Evaluation of XML Retrieval

Evaluating XML retrieval systems is essential to measure their performance in accurately retrieving relevant XML elements or documents.

Unlike traditional IR, XML retrieval requires evaluation methods that account for both content relevance and structural relevance of the results.

It uses standardized test collections like INEX (Initiative for the Evaluation of XML Retrieval), which provides XML datasets, queries, and relevance judgements for testing and comparison.

INEX 2002 defined component coverage and topical relevance as orthogonal dimensions of relevance.

The component coverage dimension evaluates whether the element retrieved is structurally correct, that is, neither too low nor too high in the tree.

It can be distinguished by four cases: -

- Exact coverage (E): The information sought is the main topic of the component and the component is a meaningful unit of information.
- Too small (S): The information sought is the main topic of the component, but the component is not a meaningful unit of information.
- Too large (L): The information sought is present in the component but is not the main topic.
- No coverage (N): The information sought is not a topic of the component.

The topical relevance dimension also has four levels:

- Highly relevant (3).
- Fairly relevant (2).
- Marginally relevant (1).
- Nonrelevant (0).

Thus, by combining content-based and structure-aware metrics, XML retrieved evaluation provides a comprehensive assessment of system effectiveness.

Text-Centric vs. Data-Centric XML Retrieval

XML documents are typically categorized into text-centric and data-centric types based on their primary purpose and structure.

The retrieval approach for these types of XML documents differs due to the nature of the data.

Text Centric XML Retrieval: -

- Text-centric documents primarily contain large blocks of unstructured and semi-structured text, annotated with XML tags to provide structure.
- The tags are used for metadata or organizational purposes, not primarily for data representation.
- Text-centric approaches are appropriate for data that are essentially text documents, marked up as XML to capture document structure.
- This is becoming a De facto standard for publishing text databases since most text documents have some form of interesting structure – paragraphs, sections, footnotes, etc.
- Examples includes assembly manuals, issues of journals, newswire articles, etc.

Data Centric XML Retrieval: -

- Data-centric approaches are commonly used for data collections with complex structures that mainly contain non-text data.
- It is often used for data interchange between systems or as a database-like format.
- Here the content is usually small and tightly coupled with the structural elements.

- A text-centric retrieval engine will have a hard time with proteomic data in bioinformatics or with the representation of a city map that forms a navigational database.
- Examples includes, Banking transactions, product catalogs, and inventory records.

Difference Between Text-Centric and Data-Centric XML Retrieval

Aspect	Text-Centric XML Retrieval	Data-Centric XML Retrieval
Definition	Focuses on large blocks of unstructured or semi-structured text.	Focuses on highly structured and schema-defined data.
Purpose	Primarily for managing and retrieving textual content.	Used for structured data storage and precise querying.
Data Type	Semi-structured or unstructured text data.	Structured data with clear schema definitions.
Examples	Books, articles, digital libraries.	Product catalogs, banking records, transaction logs.
Primary Focus	Emphasizes textual content over structure.	Emphasizes structure and content equally.
Query Type	Keyword-based or semantic queries.	Structural and content-specific queries (e.g., XPath, XQuery).
Techniques Used	Vector Space Model (VSM), semantic weighting.	Indexing, structural matching, XPath, and XQuery.
Challenges	Balancing content relevance with structural tags.	Precise matching of structure and data.
Tag Usage	Tags are metadata or used to organize text.	Tags are schema elements representing specific data fields.
Processing Complexity	Involves text processing and semantic analysis.	Requires structural parsing and relational data querying.
Example XML	Text-heavy, e.g., <code><content></code> tags with paragraphs.	Structured, e.g., <code><product></code> with <code><price></code> and <code><id></code> .
Use Cases	Digital libraries, news archives.	Inventory systems, e-commerce product searches.

Recommendation System

Recommender systems are a way of suggesting like or similar items and ideas to a user's specific way of thinking.

Recommender systems are widely used on web for recommending products and services to users.

Recommender systems try to automate aspects of a completely different information discovery model, where people try to find other people with similar tastes and then ask them to suggest new things.

The goal of recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them.

Suggestions for books on Amazon, or movies on Netflix, are real-world examples of the operation of industry strength of recommender systems.

Recommendation systems are a key part of almost every modern consumer website. The systems help drive customer interaction and sales by helping customers discover products and services they might never find themselves.

Recommender systems predict the preference of the user for these items, which could be in the form of a rating or response.

When more data becomes available for a customer profile, the recommendation becomes more accurate.

Thus, the systems are widely used in e-commerce, entertainment platform, and content delivery services.

In general, there are three types of recommender systems: -

- Collaborative recommender system is a system that produces its result based on past ratings of user with similar preferences.
- Content based recommender system is a system that produces its results based on the similarity of the content of the documents or items.
- Knowledge based recommender system is a system that produces its results based on additional and means-end knowledge.

Different types of Recommendation Techniques are: -

- Collaborative Filtering
- Content-based Filtering

Collaborative Filtering

Collaborative Filtering is a popular recommendation technique that relies on user-item interaction to suggest relevant items to users.

It assumes that if a user has agreed on certain items in the past, they will likely agree on other similar items in future.

It is a method of making automatic predictions about the interests of a single user by collecting preferences or taste information from many users.

Collaborative filtering uses the given rating data from many users for many items as the basis of predicting missing rating or creating a top-N recommendation list for a given user, i.e. the active user.

The problem of collaborative filtering is to predict how well a user will like an item that he has not rated given a set of historical preference judgments for a community of users.

Types of Collaborative Filtering are: -

1. User-based Collaborative Filtering

- Here it compares target user's preferences with other users.
- Identifies neighbor users who have similar tastes and recommends the items that those neighbors liked but the target user has not interacted with.
- Thus, here assumption is that the users with similar preferences will rate item similarly.
- The neighborhood is defined in terms of similarity between users, either by taking a given number of most similar users (k nearest neighbors) or all users within a given similarity threshold.
- Popular similarity measures for CF are the Pearson correlation and the Cosine similarity.
- User-based CF is a memory-based algorithm which tries to mimic word-of-mouth by analyzing rating data from many individuals.
- The two main problems of user-based CF are that the whole user database has to be kept in memory and that expensive similarity computation between the active user and all other users in the database has to be performed.
- For e.g. User A and User B both like movies X and Y. User B also likes movie Z. Recommend movie Z to User A.

2. Item-based Collaborative Filtering

- Here items are compared rather than users.
- It identifies items similar to those the user has liked or interacted and recommends them.
- It is a model based on approach which provides recommendations based on the relationships between items inferred from the rating matrix.
- The assumption behind this approach is that the user will prefer items that are similar to other items they like.
- This algorithm works by building an item-to-item similarity matrix which defines the relationship between pairs of items.

- Item-based CF is relatively more efficient than user-based CF since model is relatively small and can be fully pre-computed.
- It has become popularized due to its use by YouTube and Amazon to provide recommendations to users.

Content-Based Recommendation of Documents and Products

Content-Based Recommendation is a recommendation approach that suggests items to users based on the features of the items and the user's past interactions.

It relies on the assumption that if a user liked an item in the past, they will likely prefer items with similar features.

Each item is described using its features such as, keywords, topics, authors, etc.

A profile is built for each user based on the features of the items they have interacted with. For e.g. rated, purchased, or viewed. The profile represents the user's preferences.

Recommendations are generated by finding the items that are most similar to the user's profile.

Techniques like cosine similarity, Euclidean distance, or TF-IDF for text and data are used.

Steps in Content-based Recommendation are: -

- **Data collection:** Collect data on items and user interactions with items.
- **Feature Extraction:** Identifies relevant attributes of the items. We can use Natural language processing to extract keywords or topics or use structured attributes like price, category, specifications, etc.
- **Profile Building:** Create a profile for user based on their interaction history. For e.g. If a user reads many articles about "AI," their profile will emphasize keywords like "machine learning," "neural networks," etc.
- **Recommendation Generation:** Compare the user's profile with the features of all the items in the system and recommend users the items with highest similarity score.

Components of Content-based representation system architecture are: -

Information source: This is the origin of the data about the items, such as product databases, articles, or user feedback.

Item Description: These are the features or attributes of the items extracted from the information source.

Content Analyzer: Processes the item descriptions to create a structured representation (e.g., feature vectors) of the items.

Structured Item representations: Represents the processed items in a standardized format, such as vectors of features.

Profiler learner: Builds and updates user profiles based on their interactions, preferences, and feedback.

Profile Cleaner: Ensures that user profiles remain accurate by removing outdated or irrelevant preferences.

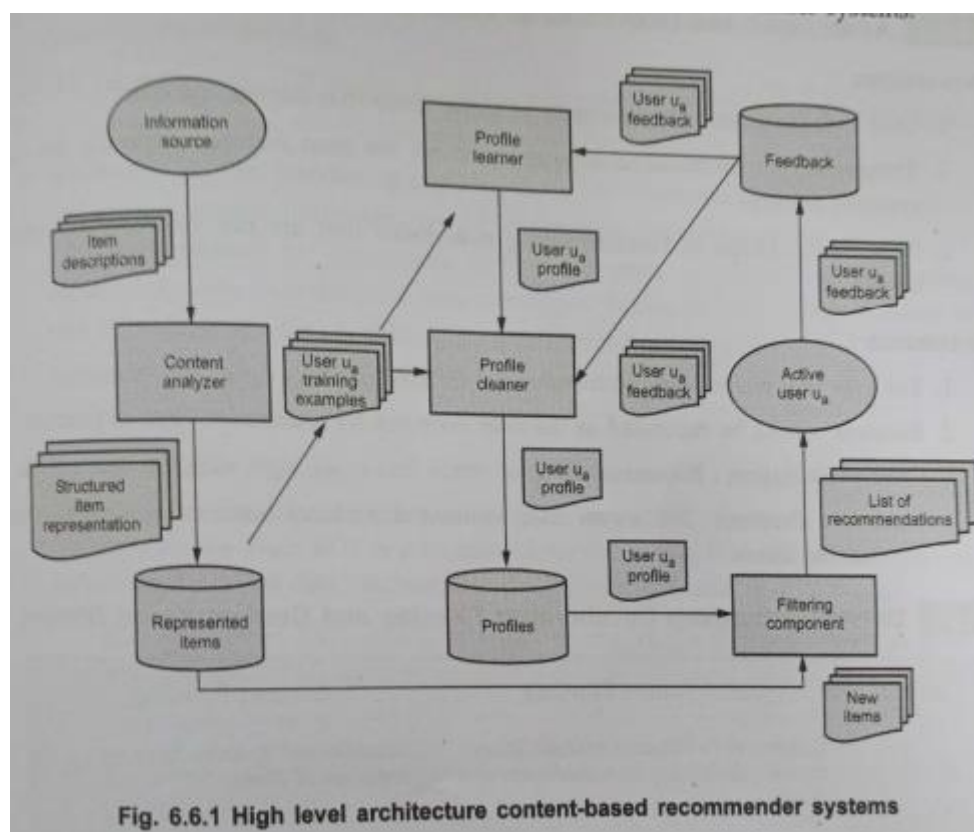
User Feedback: Collected from the user to improve the recommendation system.

Profiles: A repository of user profiles, storing user preferences and interaction data.

Filtering Components: Matches user profiles with the structured item representations to generate recommendations.

List of Recommendations: A personalized list of items presented to the user based on their profile.

Feedback loop: Users' actions and responses to recommendations are used to update profiles and improve the system's accuracy.



To overcome the limitations of content-based recommendation, hybrid approaches combine collaborative filtering and content-based methods to improve performance.

For instance, Netflix and Amazon often use hybrid models to recommend movies or products.

Difference between Collaborative filtering and content-based filtering.

Aspect	Collaborative Filtering System	Content-Based Filtering System
Basis of Recommendation	Based on user-item interactions and similarities between users.	Based on the characteristics of items and the user's preferences.
Data Dependency	Relies on user ratings, reviews, or usage patterns.	Relies on item attributes or features.
Cold Start Problem	Faces challenges when there are no prior user interactions (new users/items).	Faces challenges when item attributes are poorly defined.
Personalization	Considers preferences of similar users for recommendations.	Focuses solely on the individual user's past preferences.
Scalability	Requires a large number of users to perform effectively.	Works well even with a small user base.
Diversity of Recommendations	Can suggest items outside a user's usual preferences.	Limited to items similar to those the user has interacted with.
Domain Knowledge	Does not require detailed item information or attributes.	Requires detailed information about item features.
Adaptability	Adapts to changing user behavior based on collaborative data.	Adapts based on user's direct feedback and interactions.
Examples	Movie recommendations based on similar users' ratings.	Suggesting books similar to those previously read by the user.
Common Use Cases	Social media, e-commerce platforms with large user bases.	Niche recommendation systems like online learning platforms.

Semantic Web

1. Definition: Semantic Web is an extension of the World Wide Web that enables machines to understand and interpret the meaning (semantics) of data on the web.

2. **Objective:** It aims to create a web of data that can be processed by both humans and machines for enhanced accessibility and automation.
3. **Core Technologies:** Utilizes technologies such as RDF (Resource Description Framework), OWL (Web Ontology Language), and SPARQL (Query Language) to represent, share, and query structured data.
4. **Knowledge Representation:** Information is stored in a structured format as triples (subject, predicate, object), making it easier for machines to infer relationships.
5. **Ontology:** Plays a central role in organizing data by defining the relationships and categories within a specific domain, enabling meaningful data interconnection.
6. **Interoperability:** Facilitates seamless data sharing across different platforms, applications, and domains through standardized formats.
7. **Automation:** Enhances automation of tasks like personalized recommendations, data aggregation, and intelligent search by utilizing structured knowledge.
8. **Applications:**
 - Intelligent search engines (e.g., Google Knowledge Graph).
 - Healthcare systems for integrating patient data.
 - E-commerce for personalized product recommendations.
9. **Key Features:**
 - Machine-readability: Data is in a format machine can process.
 - Data Linking: Enables linking related data across websites and systems.
 - Context Understanding: Provides meaning to data, not just keywords.
10. **Challenges:**
 - Complexity in creating and maintaining ontologies.
 - Scalability issues for massive datasets.
 - Standardization and interoperability concerns.
11. **Benefits:** Improves data discoverability, simplifies decision-making, enhances web services, and enables intelligent agents to function effectively.
12. **Real-world Example:** Amazon uses semantic data to recommend products based on user preferences and relationships between items.
13. **Future of the Web:** Envisions a web where both structured and unstructured data is interconnected, creating a robust foundation for artificial intelligence systems.

Notes by Sahil Publication

Let's score together...