Definition

Parallel Query Processing refers to the technique of executing a user's search query **on multiple processors or machines at the same time** to speed up information retrieval. It is essential when document collections are large and many users search simultaneously.

Introduction

Search engines often store billions of documents, making single-processor searching too slow. Parallel processing solves this by distributing the workload across different nodes. Each node processes part of the query or part of the data. The results are then merged to create a single ranked output for the user. This makes search fast, scalable, and efficient.

Components / Types of Parallel Processing

1. Document Partitioning (Sharding)

- **Definition:** Documents are divided across different machines.
- **Purpose:** Each machine builds its own local index.
- **Working:** Query is broadcast to all nodes → each node returns results → they are merged.
- **Example:**
  Term "AI" searched → Node1 searches Docs 1–1000, Node2 searches Docs 1001–2000.

2. Term Partitioning

- **Definition:** Vocabulary/terms are divided across machines.
- **Purpose:** Each node stores postings lists for a set of terms.
- **Working:** Query terms go to their respective nodes → partial results combined.
- **Example:**
  Node1 stores terms A–M, Node2 stores N–Z.

3. Hybrid Partitioning

- **Definition:** Combination of document and term partitioning.
- **Purpose:** Used in large-scale search engines for balanced performance.

Working / Steps in Parallel Query Processing

1. **Query Distribution:**
   Query is forwarded to all nodes or specific nodes depending on the partitioning.
2. **Local Processing:**
   Each node retrieves matches and computes local rankings (TF-IDF, BM25).
3. **Partial Results Collection:**
   Each node sends top results back to a central coordinator.
4. **Result Merging:**
   Coordinator merges and re-ranks all results.
5. **Final Output:**
   User receives a single sorted list of relevant documents.

Advantages

- **High Speed:** Queries are processed in parallel, reducing response time.
- **Scalability:** New nodes can be added easily as data grows.
- **Fault Tolerance:** System continues to work even if one node fails.
- **High Throughput:** Can handle millions of queries per second.
- **Efficient for Large Data:** Ideal for web-scale retrieval.

Disadvantages

- **High Communication Overhead:** Merging results across nodes is costly.
- **Complex Merging Logic:** Ranking combination can be difficult.
- **Load Balancing Needed:** Uneven data distribution slows performance.
- **Infrastructure Cost:** Requires multiple machines.

Applications

- Web search engines (Google, Bing).
- Large enterprise search systems.
- Digital libraries with huge datasets.
- Recommendation systems and real-time analytics.
- E-commerce search (Amazon, Flipkart).

Example

Searching “machine learning”:

- Node1 processes documents 1–5000
- Node2 processes 5001–10,000
- Node3 processes 10,001–15,000
  All nodes send top results → coordinator merges → returns final output.

✅ MapReduce in Information Retrieval

MapReduce is a distributed programming framework used for processing and generating large datasets. In IR, it is used to build indexes, compute statistics, and analyze text across many machines efficiently.

Introduction

Modern IR systems need to analyze billions of documents, which cannot be handled by a single machine. MapReduce solves this by breaking the task into two functions—Map and Reduce—and executing them in parallel on multiple nodes. It automatically manages data distribution, fault tolerance, and aggregation, making it ideal for large-scale indexing.

Components of MapReduce

1. Map Function

- **Definition:** Reads input documents and emits key–value pairs.
- **Purpose:** Extract useful information (like terms and docIDs).
- **Example Output:** (“AI”, Doc3)

2. Shuffle & Sort Phase

- **Definition:** The system groups all values belonging to the same key.
- **Purpose:** Prepares data for reduction.
- **Example:**
  (“AI”, Doc3), (“AI”, Doc7), (“AI”, Doc8) → grouped together.

3. Reduce Function

- **Definition:** Collects all values for a term and combines them into a postings list.
- **Purpose:** Builds the inverted index entries.
- **Example Output:**
  “AI” → [Doc3, Doc7, Doc8]

Working / Steps in MapReduce for IR

1. **Input Splitting:**
   Large document collection divided into blocks.
2. **Map Phase:**
   Each block processed to emit (term, docID).
3. **Shuffle Phase:**
   System groups all pairs for the same term.
4. **Reduce Phase:**
   Term → list of all documents containing that term.
5. **Final Output:**
   Complete inverted index stored on disk.

Advantages

- **Highly Scalable:** Runs on hundreds or thousands of nodes.
- **Automatic Fault Tolerance:** Re-executes failed tasks.
- **Simple Programming Model:** Only Map and Reduce functions needed.
- **Efficient for Large Datasets:** Ideal for billions of documents.
- **Low Cost:** Uses commodity hardware.

Disadvantages

- **Not Real-Time:** High latency; not suitable for online search queries.
- **Slow for Iterative Algorithms:** Writes to disk after each stage.
- **Requires Large Cluster:** Best performance seen only on big systems.
- **Batch Oriented:** Cannot handle continuous updates efficiently.

Applications

- Constructing inverted indexes for search engines.
- Calculating TF, DF, and IDF values.
- Analyzing query logs and click data.
- Generating n-grams and co-occurrence statistics.
- Computing PageRank.
- Cleaning and preprocessing large text corpora.

Example

For the document:
“AI improves data science.”

Map Output:

- (“AI”, Doc1)
- (“improves”, Doc1)
- (“data”, Doc1)
- (“science”, Doc1)

Reduce Output:

- AI → [Doc1]
- data → [Doc1]
- science → [Doc1]

This forms part of the inverted index.

Web Retrieval: Search Engine Architectures (10-Marks Answer)

Introduction (5 lines)

Web Retrieval is the process of finding relevant web pages from the huge and unstructured World Wide Web.
Search Engine Architecture describes the internal design of a search engine used to crawl, store, index, and rank web documents.
It is needed because the web grows continuously, and efficient retrieval requires structured components.
The architecture ensures fast, scalable, and accurate search results.
Its working involves crawling websites, indexing pages, ranking them, and returning the most relevant results.

Main Components of a Search Engine

1. Web Crawler (Spider/Robot)

Definition: A program that automatically visits web pages and collects their content.
Purpose: Discovers new, updated, or linked pages on the internet.
Working Step (one line): Starts from seed URLs and follows hyperlinks to download pages.

2. Document Repository

Definition: A storage area where all downloaded webpages and metadata are kept.
Purpose: Acts as the main database for raw HTML pages before indexing.
Working Step (one line): Stores fetched content along with URL, timestamp, and metadata.

3. Indexer

Definition: Component that processes documents to create searchable indexes.
Purpose: Converts raw text into structures like inverted indexes for fast retrieval.
Working Step (one line): Performs tokenization, stop-word removal, stemming, and builds term–document lists.

4. Query Processor

Definition: Module that interprets and analyzes user queries.
Purpose: Matches user queries with indexed terms for accurate retrieval.
Working Step (one line): Parses the query, expands it if needed, and sends it to the ranking engine.

5. Ranking Engine

Definition: System that decides the order of results based on relevance.
Purpose: Ensures the most useful and trustworthy pages appear first.
Working Step (one line): Applies ranking formulas such as TF-IDF, BM25, or PageRank.

6. User Interface (Search UI)

Definition: The front-end where users enter queries and view results.
Purpose: Displays ranked results, snippets, and related information clearly.
Working Step (one line): Presents results retrieved from the ranking engine to the user.

Overall Working Process (Steps)

1. Crawling – Collecting web pages from the internet.
2. Storing – Saving pages in the document repository.
3. Indexing – Converting documents into searchable structures.
4. Query Processing – Understanding the user’s search request.
5. Ranking – Ordering results based on relevance.
6. Retrieval – Displaying the final ranked list to the user.

Advantages

- Fast retrieval due to structured indexes.
- Scalable and supports billions of documents.
- High relevance through ranking techniques.
- Continuous updates maintain freshness of data.

Disadvantages

- Requires high computational and storage resources.
- May index spam or low-quality pages.
- Does not cover the entire web (deep web limitation).
- Potential privacy concerns due to user data collection.

Applications

- General web search engines (Google, Bing).
- E-commerce search (Flipkart, Amazon).
- Academic search (Google Scholar).
- News and media search systems.
- Enterprise and intranet search engines.
- Social media search (Twitter, Facebook).

Cluster-Based Architecture (10-Marks Answer)

Introduction (5 Lines)

Cluster-Based Architecture is an approach in web retrieval where documents are grouped into clusters based on similarity before searching.
It is used because searching through the entire document collection is slow, especially when the dataset is very large.
By organizing similar documents into clusters, search engines can limit the search space and improve retrieval speed.
This architecture helps retrieve more relevant results by focusing on the most related clusters.
Its working includes document clustering, cluster indexing, query–cluster matching, and retrieval from selected clusters.

Main Components of Cluster-Based Architecture

1. Document Clusters

Definition: Groups of documents that share similar content or topics.
Purpose: Reduces search space by grouping related documents together.
Working (one line): Documents are clustered using algorithms like K-Means or Hierarchical Clustering.

2. Cluster Representatives (Centroids / Leaders)

Definition: A summary or representative document that reflects the main theme of a cluster.
Purpose: Used to compare user queries and quickly identify the closest cluster.
Working (one line): System computes a centroid based on term weights of documents in that cluster.

3. Cluster Index

Definition: An index that stores cluster-level information instead of indexing individual documents.
Purpose: Allows the system to select the most relevant clusters first before document-level search.
Working (one line): Contains representative terms, cluster size, and pointers to member documents.

4. Query–Cluster Matching Module

Definition: Component that determines which clusters are most relevant to a given query.
Purpose: Narrows down the search to a few highly relevant clusters.
Working (one line): Matches query terms with cluster representatives using cosine similarity or TF-IDF.

5. Intra-Cluster Search Engine

Definition: Performs detailed searching inside the selected clusters.
Purpose: Provides accurate document-level retrieval after cluster filtering.
Working (one line): Applies ranking functions like TF-IDF or BM25 to documents within chosen clusters.

6. User Interface

Definition: The part where users enter queries and view final ranked results.
Purpose: Presents clean and refined results retrieved from selected clusters.
Working (one line): Displays ranked documents from the best-matching clusters.

Working Process (Steps)

1. Document Collection – The system gathers documents from the web.
2. Clustering – Documents are grouped into clusters based on similarity.

3. Cluster Indexing – Representatives or centroids are created for each cluster.
4. Query Processing – User query is analyzed and processed.
5. Cluster Selection – The most relevant cluster(s) are chosen based on similarity.
6. Intra-Cluster Retrieval – Detailed search is done within selected clusters.
7. Ranking – Documents are ranked based on relevance.
8. Display – Final results are shown to the user.

Advantages

- Reduces search time by limiting retrieval to a few clusters.
- Improves relevance because results come from topic-focused groups.
- Supports scalable and large datasets.
- Helps in topic-based browsing and organization of documents.
- Efficient for exploratory search (when users are not sure what exactly they want).

Disadvantages

- Clustering large datasets is computationally expensive.
- Poor clustering may lead to irrelevant results.
- Hard to maintain when documents update frequently.
- Requires careful selection of clustering algorithm and number of clusters.

Applications

- Web search engines for topic-based retrieval.
- News classification and topic grouping.
- Digital libraries and academic search.
- E-commerce product clustering (similar items grouped).
- Recommendation systems based on user interests.
- Organization of large document repositories.

Distributed Architectures in Web Retrieval (10-Marks Answer)

Introduction (5 Lines)

Distributed Architecture refers to a design where a search engine's tasks are divided across **multiple machines or servers** instead of a single system.
It is used because web-scale data is massive and cannot be processed efficiently on one machine.
By distributing crawling, storage, indexing, and query processing, the system achieves faster retrieval and higher reliability.
Distributed architecture ensures scalability, fault tolerance, and continuous availability of search services.
Its working involves dividing documents or indexes, parallel processing, and merging results for the user.

Main Components of Distributed Architecture

1. Distributed Crawlers

Definition: Multiple crawlers running on different machines to fetch web pages simultaneously.
Purpose: Increases crawling speed and ensures coverage of a larger web space.
Working (one line): Each crawler is assigned a portion of the web to fetch and store pages independently.

2. Distributed Document Repository

Definition: Storage system spread across multiple servers to hold crawled content.
Purpose: Handles very large datasets efficiently and allows parallel access.
Working (one line): Each server stores a subset of the documents for later indexing.

3. Distributed Indexer

Definition: Indexing system that builds inverted indexes in a distributed manner.
Purpose: Enables scalable and faster indexing of massive datasets.
Working (one line): Each server processes its local documents to create partial indexes that are later merged.

4. Query Processor

Definition: Module that processes user queries across distributed indexes.
Purpose: Ensures that queries are handled efficiently and in parallel.
Working (one line): Query is broadcast to all index nodes, each returns top results.

5. Result Merging and Ranking Engine

Definition: Combines partial results from multiple servers and ranks documents.
Purpose: Produces a final, globally ranked list for the user.
Working (one line): Uses ranking algorithms like TF-IDF, BM25, or PageRank on merged results.

6. User Interface (Search UI)

Definition: Front-end for users to enter queries and view results.
Purpose: Displays aggregated, ranked results from distributed servers.
Working (one line): Presents the final merged list of relevant documents clearly.

Working Process (Steps)

1. Crawling – Distributed crawlers fetch web pages in parallel.
2. Storage – Pages are stored across multiple servers.
3. Indexing – Each server builds partial inverted indexes.
4. Query Processing – User query sent to all index nodes.
5. Result Merging – Partial results combined and ranked.
6. Retrieval – Final ranked results delivered to the user.

Advantages

- **High Scalability:** Can handle web-scale data by adding more servers.
- **Faster Retrieval:** Parallel processing reduces query response time.
- **Fault Tolerance:** Failure of one server does not stop the system.
- **Load Distribution:** Efficient utilization of resources across servers.
- **Supports Real-Time Updates:** Easier to update indexes in parts without rebuilding the entire index.

Disadvantages

- **Complex System Management:** Needs careful monitoring and synchronization.
- **Communication Overhead:** Merging results from multiple nodes increases network load.
- **Consistency Issues:** Updates must be propagated to multiple nodes.
- **Higher Infrastructure Cost:** Requires multiple machines and storage systems.

Applications

- Web search engines (Google, Bing) for global-scale search.
- Distributed digital libraries and archives.
- Large-scale e-commerce platforms for fast product search.
- Enterprise search systems across multiple servers.
- News aggregation and topic-based retrieval.
- Social media search and recommendation systems.

1. Search Engine Ranking (10-Marks Answer)

Introduction (5 Lines)

Search Engine Ranking is the process of ordering web pages returned for a query based on their relevance and usefulness.
It ensures that users see the most important and relevant results at the top.
Ranking considers factors such as content quality, user behavior, freshness, and popularity.
Search engines use ranking algorithms to evaluate these factors and assign scores to pages.
The working involves analyzing both the content and context of pages to determine their position in search results.

Main Components / Concepts

1. Relevance Scoring

- **Definition:** Evaluating how well a page's content matches the user query.
- **Purpose:** Ensures results answer the user's intent.
- **Working (one line):** Compares query terms with document terms using models like TF-IDF or BM25.

2. Popularity / Authority

- **Definition:** Measures how well-known or trusted a page is.
- **Purpose:** Pages referenced by many users or other pages are considered important.
- **Working (one line):** Higher links, citations, or mentions increase authority.

3. Freshness / Timeliness

- **Definition:** How recently a page was created or updated.
- **Purpose:** Ensures current and relevant information appears first.
- **Working (one line):** Newer pages may receive higher ranking for time-sensitive queries.

4. User Interaction Signals

- **Definition:** Click-through rates, dwell time, and bounce rates.
- **Purpose:** Helps measure actual usefulness to users.
- **Working (one line):** Pages with higher engagement are ranked higher.

Working Process (Steps)

1. Crawling – Pages are fetched from the web.
2. Indexing – Documents are processed and stored for retrieval.
3. Scoring – Each page is assigned a relevance and importance score.
4. Ranking – Pages are sorted based on combined scores.
5. Retrieval – Top results are displayed to the user.

Advantages

- Provides relevant and useful results to users.
- Improves user satisfaction and search efficiency.
- Supports personalization based on user behavior.
- Can combine multiple factors (content, links, freshness).

Disadvantages

- Can be influenced by spam or SEO manipulation.
- May favor popular pages over highly relevant but less known pages.
- Computationally intensive for large web datasets.
- Continuous updates are required to maintain accuracy.

Applications

- Google, Bing, and other web search engines.
- E-commerce product ranking.
- Academic paper ranking (Google Scholar).
- News and media content ranking.
- Recommendation systems based on page importance.

2. Link-Based Ranking (10-Marks Answer)

Introduction (5 Lines)

Link-Based Ranking is a ranking method that evaluates web pages based on links pointing to them.
It assumes that a page referenced by many high-quality pages is more authoritative.
This method is essential because content alone may not reflect a page's importance.
It uses algorithms like PageRank and HITS to analyze the web's link structure.
Its working involves assigning scores to pages based on incoming links and linking page quality.

Main Components / Concepts

1. Backlinks / Incoming Links

- **Definition:** Links from other pages pointing to a web page.
- **Purpose:** Indicate popularity or authority.
- **Working (one line):** Pages with more high-quality backlinks receive higher scores.

2. PageRank Algorithm

- **Definition:** Evaluates the importance of a page based on its incoming links.
- **Purpose:** Measures global authority using the web graph.

- **Working (one line):** Score of a page depends on the scores of pages linking to it divided by their outgoing links.

Formula (simplified):
PR(A) = (1-d) + d * Σ [PR(Pi)/C(Pi)]

3. HITS Algorithm (Hub and Authority)

- **Definition:** Identifies authoritative pages and hubs for specific topics.
- **Purpose:** Authority pages provide content, hubs link to authorities.
- **Working (one line):** Authority score = sum of hub scores pointing to it; hub score = sum of authority scores it points to.

4. Link Quality Assessment

- **Definition:** Evaluates relevance and trustworthiness of linking pages.
- **Purpose:** Reduces influence of spam or low-quality links.
- **Working (one line):** High-quality links contribute more to ranking than low-quality links.

Working Process (Steps)

1. Crawling – Collect pages and links from the web.
2. Indexing – Build link structure database.
3. Link Analysis – Count and evaluate backlinks.
4. Score Calculation – Compute PageRank or HITS scores.
5. Ranking – Sort pages based on link-based importance.
6. Retrieval – Present top-ranked pages to the user.

Advantages

- Identifies authoritative and trusted pages.
- Improves result quality beyond content relevance.
- Resistant to keyword stuffing or content spam.
- Useful for topic-specific searches (HITS).

Disadvantages

- Vulnerable to link manipulation (link farms).
- New pages start with low ranking until links accumulate.
- Computationally intensive for large web graphs.
- May not consider content relevance fully without combination.

Applications

- Google Search (PageRank-based ranking).
- Academic citation ranking (papers with more citations).
- Social media influencer scoring (pages/users with more backlinks).
- Topic-specific search engines (HITS algorithm).
- E-commerce product popularity ranking using references or reviews.

Page Ranking Algorithms (10-Marks Answer)

Introduction (5 Lines)

Page Ranking Algorithms are methods used by search engines to assign a score to each web page based on its importance and relevance.
They are used to order search results so that the most authoritative and useful pages appear first.
Ranking considers both **content relevance** and **link-based importance**.
These algorithms help users find high-quality information quickly and efficiently.
Common algorithms include **PageRank**, **HITS**, and modern hybrid ranking models.

Main Page Ranking Algorithms

1. PageRank Algorithm

Definition: A link-based algorithm that evaluates a page's importance based on incoming links.
Purpose: Pages referenced by many high-quality pages are considered more authoritative.
Working (one line): The score of a page depends on the scores of linking pages divided by their outgoing links.

Formula (simplified):
PR(A) = (1-d) + d * Σ [PR(Pi)/C(Pi)]

- PR(A) = PageRank of page A
- d = damping factor (~0.85)

- Pi = pages linking to A
- C(Pi) = number of outgoing links from Pi

Example:
Page A has incoming links from Page B and C. The PageRank of B and C contributes to A's rank.

2. HITS Algorithm (Hyperlink-Induced Topic Search)

Definition: Classifies pages as **authorities** and **hubs** for a specific topic.
Purpose: Authority pages provide content; hub pages link to authorities.
Working (one line):

- Authority score = sum of hub scores linking to it
- Hub score = sum of authority scores it points to

Example:
A directory page linking to multiple research papers is a hub; the papers themselves are authorities.

3. Weighted PageRank

Definition: An enhancement of PageRank that considers link quality, not just quantity.
Purpose: Pages with more important inbound links are ranked higher.
Working (one line): Assigns weights to links based on the rank of the source page.

4. Topic-Sensitive PageRank

Definition: Calculates PageRank based on topic-specific importance.
Purpose: Gives higher scores to pages that are authoritative for particular topics.
Working (one line): Maintains separate PageRank scores for each topic category.

5. Personalized / User-Based Ranking

Definition: Ranking adjusted according to a user's preferences or search history.
Purpose: Improves relevance for individual users.
Working (one line): Combines global PageRank with user interest profiles.

Working Process (Steps)

1. Crawling – Collect web pages and links.
2. Indexing – Store page content and link structure.
3. Link Analysis – Analyze incoming and outgoing links.
4. Score Computation – Calculate PageRank, HITS, or weighted scores.
5. Ranking – Order pages based on calculated scores.
6. Retrieval – Display top-ranked pages for user queries.

Advantages

- Identifies authoritative and important pages.
- Reduces spam influence (especially link-based algorithms).
- Works well for large-scale web search.
- Can be combined with content relevance for better results.
- Topic-specific and personalized variants improve search accuracy.

Disadvantages

- Vulnerable to link manipulation (link farms, paid links).
- New pages may rank low until links accumulate.
- Computationally expensive for very large web graphs.
- Some algorithms do not consider content relevance fully.

Applications

- Google Search (PageRank-based ranking).
- Academic paper ranking (citation counts).
- Topic-specific search engines (HITS).
- Personalized recommendations (using user-based ranking).
- E-commerce and social media platforms for ranking products or pages.

Simple Ranking Functions and Evaluations (10-Marks Answer)

Introduction (5 Lines)

Simple Ranking Functions are basic methods used by search engines to order documents based on relevance to a user query.
They are essential for evaluating which documents should appear first in search results.
Ranking functions help measure how well a document matches a query using straightforward calculations.
Evaluation methods assess the effectiveness of these ranking functions in returning relevant results.
These concepts form the foundation before using advanced algorithms like PageRank or BM25.

Simple Ranking Functions

1. Term Frequency (TF)

- **Definition:** Measures how many times a query term appears in a document.
- **Purpose:** Documents with higher frequency of query terms are considered more relevant.
- **Working (one line):** TF = Number of times term t appears in document d.

Example:
Query = "AI", Document = “AI is the future of AI research” → TF(AI) = 2.

2. Inverse Document Frequency (IDF)

- **Definition:** Measures how rare a term is across all documents.
- **Purpose:** Rare terms contribute more to relevance than common terms.
- **Working (one line):** IDF(t) = log(Total documents / Number of documents containing t).

3. TF-IDF

- **Definition:** Combines TF and IDF to give a weighted score for terms.
- **Purpose:** Balances term frequency with term uniqueness to improve relevance.
- **Working (one line):** TF-IDF(t,d) = TF(t,d) × IDF(t).

4. Boolean Ranking

- **Definition:** Simple yes/no ranking based on query term presence.
- **Purpose:** Returns documents containing the query terms only.
- **Working (one line):** If document contains all query terms → relevant, else → not relevant.

5. Vector Space Model (Cosine Similarity)

- **Definition:** Represents documents and queries as vectors in term space.
- **Purpose:** Measures similarity between query and documents using cosine of angle between vectors.
- **Working (one line):** Cosine similarity = (query • document) / (||query|| × ||document||).

Evaluation of Ranking Functions

1. Precision

- **Definition:** Fraction of retrieved documents that are relevant.
- **Formula:** Precision = (Relevant documents retrieved) / (Total documents retrieved)
- **Purpose:** Measures accuracy of search results.

2. Recall

- **Definition:** Fraction of relevant documents that are retrieved.
- **Formula:** Recall = (Relevant documents retrieved) / (Total relevant documents)
- **Purpose:** Measures completeness of search results.

3. F-Measure (F1 Score)

- **Definition:** Harmonic mean of precision and recall.
- **Formula:** F1 = 2 × (Precision × Recall) / (Precision + Recall)
- **Purpose:** Balances precision and recall in one metric.

4. Mean Average Precision (MAP)

- **Definition:** Average precision over multiple queries.
- **Purpose:** Evaluates overall effectiveness of ranking functions across a set of queries.

5. Discounted Cumulative Gain (DCG)

- **Definition:** Measures usefulness of documents based on their position in ranking.

- **Purpose:** Higher-ranked relevant documents contribute more to the score.
- **Working (one line):** DCG = Σ (relevance / log2(rank+1))

Working Process (Steps for Evaluation)

1. Run a query on the search system.
2. Retrieve and rank documents using a ranking function.
3. Compare retrieved results with ground truth of relevant documents.
4. Calculate metrics like Precision, Recall, F1, MAP, or DCG.
5. Analyze and improve ranking function based on evaluation.

Advantages

- Simple functions are easy to implement and understand.
- Provide baseline for more advanced ranking algorithms.
- Can quickly filter out irrelevant documents.
- Metrics allow quantitative evaluation of search effectiveness.

Disadvantages

- Cannot handle semantic meaning or context of words.
- Sensitive to synonyms and vocabulary mismatch.
- Boolean ranking may return too many or too few results.
- Advanced models are required for complex queries.

Applications

- Basic search engines and document retrieval systems.
- Academic and research paper search.
- E-commerce product filtering and ranking.
- Legal document and enterprise search.
- Evaluating new ranking algorithms for relevance testing.

Web Crawler (10-Marks Answer)

Introduction (5 Lines)

A Web Crawler is a program that automatically browses the World Wide Web to collect web pages and data.
It is the first step in web retrieval systems and search engines.
Crawlers help in creating a structured repository of web content for indexing and retrieval.
They are used to discover new pages, update existing content, and extract information.
Python provides libraries like Scrapy and Beautiful Soup to build efficient crawlers easily.

Web Crawler Structure

1. Seed URLs

- **Definition:** Initial set of web addresses from which the crawler starts its journey.
- **Purpose:** Provides a starting point to discover the rest of the web.
- **Working (one line):** Crawler fetches these URLs first and extracts links for further crawling.

2. URL Frontier / Queue

- **Definition:** A data structure that stores URLs to be crawled next.
- **Purpose:** Manages the order of URLs for breadth-first or depth-first crawling.
- **Working (one line):** Adds newly discovered URLs and removes already crawled URLs.

3. Fetcher / Downloader

- **Definition:** Component that downloads web pages from the internet.
- **Purpose:** Retrieves page content including HTML, CSS, and JavaScript.
- **Working (one line):** Sends HTTP requests and receives responses for each URL in the frontier.

4. Parser / Extractor

- **Definition:** Analyzes downloaded content and extracts links or data.

- **Purpose:** Identifies new URLs and relevant information for indexing.
- **Working (one line):** Parses HTML and extracts links using DOM or libraries like Beautiful Soup.

5. Storage / Repository

- **Definition:** Stores downloaded pages and extracted data.
- **Purpose:** Provides a persistent collection of documents for indexing or analysis.
- **Working (one line):** Saves pages in files, databases, or cloud storage.

Web Crawler Libraries

1. Scrapy (Python)

- **Definition:** A powerful Python framework for large-scale web crawling and scraping.
- **Purpose:** Allows structured, scalable, and fast crawling.
- **Features:** Supports pipelines, middleware, automatic throttling, and multi-threaded crawling.
- **Working (one line):** Define spiders to fetch URLs, parse content, and store results.

2. Beautiful Soup (Python)

- **Definition:** A Python library for parsing HTML and XML documents.
- **Purpose:** Helps extract specific data elements from web pages.
- **Working (one line):** Provides methods to search, navigate, and modify HTML elements easily.

3. Other Libraries

- **Requests:** For fetching web pages via HTTP.
- **Selenium:** For dynamic content crawling rendered by JavaScript.
- **Urllib / LXML:** For lightweight page fetching and parsing.

Applications of Web Crawlers

1. Search engines (Google, Bing) to index the web.
2. Price comparison websites and e-commerce data extraction.
3. Social media monitoring and analytics.

Advantages

- Automates web data collection efficiently.
- Can scale to crawl millions of pages.
- Enables timely updates of dynamic content.
- Supports structured extraction for analytics.

Disadvantages

- Crawling dynamic or protected websites may be challenging.
- Excessive crawling can overload servers.
- Needs careful handling of duplicate pages and infinite loops.
- Legal and ethical restrictions must be considered.

Main Challenges (Only 6, with clear explanations)

1. Massive Scale of the Web

- The web contains billions of interconnected pages.
- Storing, crawling, and indexing such huge data requires high computational and storage resources.
- Search engines cannot crawl the entire web frequently due to its enormous size.

2. Dynamic and Continuously Changing Content

- Web pages update frequently (news, e-commerce, social media).
- New pages appear and old ones disappear unpredictably.
- Maintaining up-to-date indexes is difficult because crawlers cannot revisit all pages instantly.

3. Heterogeneous and Unstructured Data

- Web content includes text, images, videos, scripts, tables, and dynamic elements.
- There is no uniform structure, making extraction and processing complex.

- Parsing multimedia and non-textual content adds more difficulty.

4. Duplicate and Near-Duplicate Content

- Many websites replicate the same articles, product descriptions, or copied content.
- Duplication wastes storage, increases crawling effort, and affects ranking quality.
- Detecting duplicates requires hashing, similarity checks, and additional computation.

5. Web Spam and Malicious Sites

- Some websites use keyword stuffing, fake backlinks, and cloaking to manipulate rankings.
- Malware and phishing pages can harm users and mislead crawlers.
- Search engines must apply filtering, classification, and link analysis to detect spam.

6. Ambiguous and Short User Queries

- Most users type very short queries (2–3 words) with multiple meanings.
- Understanding user intent becomes difficult without context.
- Search systems must use query expansion, semantic analysis, and ranking techniques to improve relevance.

Components of Focused Web Crawlers

Main Components of a Focused Web Crawler

1. Seed URLs

- These are the initial topic-relevant pages from which the crawler starts.
- They provide a strong starting point for finding more related pages.
- Seeds are selected manually or from trusted directories.

2. URL Frontier (Priority Queue)

- Stores URLs that need to be crawled next.
- Unlike general crawlers, focused crawlers use a **priority queue**, where URLs expected to be more relevant are given higher priority.
- Priority is determined using topic similarity, link structure, or relevance score.

3. Page Downloader / Fetcher

- Sends HTTP requests to download pages from the URL frontier.
- Ensures polite crawling using time gaps and robots.txt checking.
- Fetches HTML content for further analysis.

4. Content Analyzer / Page Classifier

- Examines downloaded pages to determine if they match the required topic.
- Uses methods like keyword matching, TF-IDF, Naive Bayes, or machine learning classifiers.
- If the page is relevant → it is stored and its outgoing links are processed.
- If not relevant → the page is discarded.

5. Link Extractor

- Extracts all hyperlinks from each downloaded page.
- Filters out irrelevant links such as ads, navigation bars, or duplicate pages.
- Passes extracted links to the relevance predictor.

6. Relevance Predictor / Link Classifier

- Predicts how relevant each outgoing link might be before actually downloading the page.
- Uses anchor text, surrounding text, URL patterns, and parent page relevance.
- Assigns a relevance score to each link and sends high-scoring links to the URL frontier.

7. Repository / Storage System

- Stores all relevant pages, metadata, and extracted content.
- Helps in building domain-specific search engines or datasets.
- Maintains indexes for fast retrieval and future processing.