DTS207TC Database Development and Design

Lecture 12

Chap 31. Information Retrieval

Di Zhang, Autumn 2025

Outline

- History
- IF-IDF
- Pagerank
- Revert-Index
- Web Crawler

The History of Search Engines

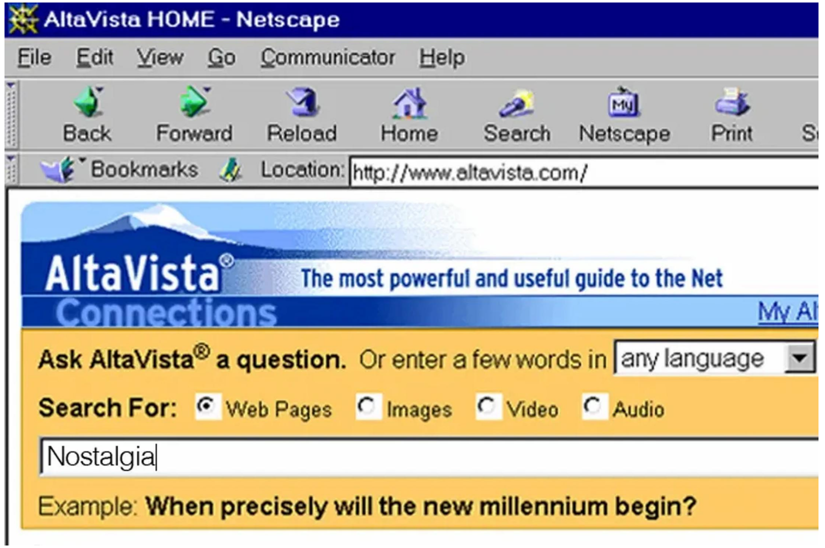
Xi'an Jiaotong-Liverpool University 西交利物浦大学

- The Early Days: Pre-Web & The First Steps
- Before Google: The Pioneers (1990-1995)
  - The Problem: The early internet was a collection of disconnected files. No easy way to find anything.
  - Archie (1990): The first "search engine." It simply indexed the names of files on public FTP servers.
  - Gopher & Veronica/Jughead: Searched menu-based systems, not full text.
  - The Web Explosion: The invention of the World Wide Web (WWW) created a urgent need for better navigation tools.
  - Wandex (1993): The first web robot and indexer, cataloging the early web.

El primer buscador nació en 1993 y es conocido como Wandex, un índice desarrollado por Mathew Gray en el MIT (Instituto de Tecnología de Massachusetts). Este buscador surgió como un proyecto de la World Wide Web.

The Rise of Web Crawlers & The First Giants

- The Crawler Revolution (1994-1998)
  - WebCrawler (1994): The first full-text search engine that allowed users to search for any word on any webpage.
  - The Directory Approach - Yahoo!: Founded by humans who manually categorized websites. It was a curated web directory, not a true crawler-based engine.
  - The Powerhouses: AltaVista & Excite:
    - AltaVista: Launched by DEC, it was fast, powerful, and supported natural language queries and advanced search operators. It was the king of search for years.
  - The Common Flaw: These early engines relied heavily on basic keyword matching, making them easy to spam with irrelevant pages.


AltaVista HOME - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location:
AltaVista®
Connections
My Al

Location: http://www.altavista.com/

The most powerful and useful guide to the Net

Ask AltaVista® a question. Or enter a few words in any language

Search For: [✓ Web Pages] Images Video Audio

Nostalgia

Example: When precisely will the new millennium begin?

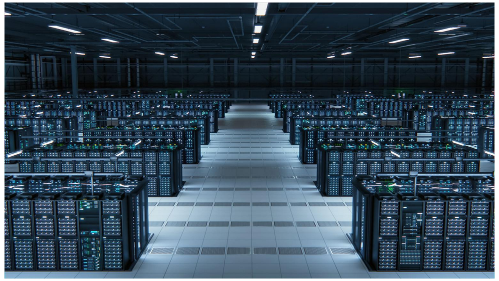
The Google Revolution: PageRank Changes Everything

The Game Changer: Google (1998-Present)

- The Breakthrough: PageRank Algorithm.
  - Founded by Larry Page and Sergey Brin at Stanford.
  - Core Idea: A webpage is important if other important pages link to it. It treated links as "votes."
- Why It Won:
  - Relevance: Delivered vastly more relevant and higher-quality results.
  - Speed: Incredibly fast.
  - Simplicity: The clean, uncluttered homepage was a stark contrast to the "portal" look of competitors.
- The Domination: Google's superior technology quickly made it the default search engine for the world.

The Modern Era & The AI-Powered Future

- Beyond Keywords: Voice, Personalization, and AI
  - Mobile & Voice Search: Search moved from the desktop to everywhere. Queries became conversational ("OK Google, where's the nearest coffee shop?").
  - Personalization: Search results are tailored to your location, search history, and preferences.
  - Featured Snippets & "Answer Engines": Google started pulling information directly to the top of the page, aiming to answer your question without a click.
- The AI & LLM Revolution (Now & Future):
  - BERT & MUM: Google's AI models understand context and nuance in queries like never before.
  - Generative AI & ChatGPT: The rise of Large Language Models (LLMs) is shifting search from a "list of links" to a conversational experience. The future is about getting synthesized, direct answers and engaging in a dialogue.



Information Retrieval Systems

- **Information retrieval (IR)** systems use a simpler data model than database systems
  - Information organized as a collection of documents
  - Documents are unstructured, no schema
- Information retrieval locates relevant documents, on the basis of user input such as keywords or example documents
  - e.g., find documents containing the words "database systems"
- Can be used even on textual descriptions provided with non-textual data such as images
- Web search engines are the most familiar example of IR systems

Information Retrieval Systems (Cont.)

- Differences from database systems
  - IR systems don’t deal with transactional updates (including concurrency control and recovery)
  - Database systems deal with structured data, with schemas that define the data organization
  - IR systems deal with some querying issues not generally addressed by database systems
    - Approximate searching by keywords
    - Ranking of retrieved answers by estimated degree of relevance

Keyword Search

- In **full text** retrieval, all the words in each document are considered to be keywords.
  - We use the word **term** to refer to the words in a document
- Information-retrieval systems typically allow query expressions formed using keywords and the logical connectives *and, or,* and *not*
  - *And*s are implicit, even if not explicitly specified
- Ranking of documents on the basis of estimated relevance to a query is critical
  - Relevance ranking is based on factors such as
    - Term frequency
      - Frequency of occurrence of query keyword in document
    - Inverse document frequency
      - How many documents the query keyword occurs in
        - Fewer ➔ give more importance to keyword
    - Hyperlinks to documents
      - More links to a document ➔ document is more important

Relevance Ranking Using Terms

- **TF-IDF** (Term frequency/Inverse Document frequency) ranking:

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

$\mathtt{tf}(t,d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

x

$\mathtt{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

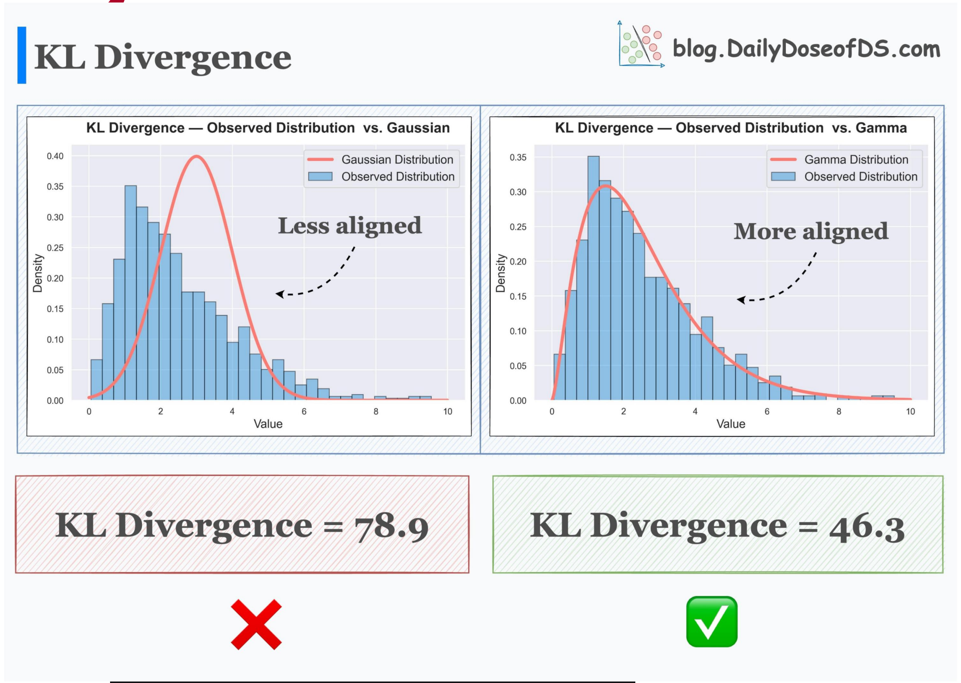
$\mathtt{tfidf}(t,d,D) = \mathtt{tf}(t,d) \cdot \mathtt{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

TF-IDF is the measure of importance in information theory*

- The Goal:
  - Find term weights that make each document stand out from the corpus average.
- The Math:
  - We model this by maximizing the difference between:
    - A term's local distribution in a doc, P(t|d) (≈ TF)
    - Its global background distribution, P(t)
- The Insight:
  - The optimal measure for this difference is KL Divergence:



$$D_{KL}(P_d||P_C) = \sum_w P_d(w) \log \frac{P_d(w)}{P_C(w)}$$

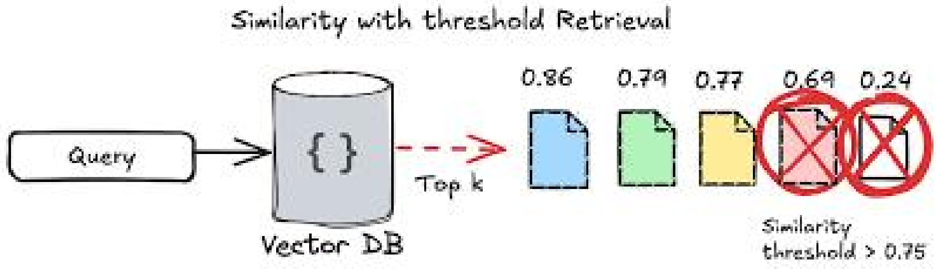
$$\text{Score}(w) \propto \underbrace{P_d(w)}_{\text{TF}} \times \underbrace{\log \frac{N}{df(w)}}_{\text{IDF}}$$

Relevance Ranking Using Terms (Cont.)

- Most systems add to the above model
  - Words that occur in title, author list, section headings, etc. are given greater importance
  - Words whose first occurrence is late in the document are given lower importance
  - Very common words such as “a”, “an”, “the”, “it” etc. are eliminated
    - Called **stop words**
  - **Proximity**: if keywords in query occur close together in the document, the document has higher importance than if they occur far apart
- Documents are returned in decreasing order of relevance score
  - Usually only top few documents are returned, not all

Similarity Based Retrieval

- Similarity based retrieval - retrieve documents similar to a given document
  - Similarity may be defined on the basis of common words
    - E.g., find *k* terms in A with highest *TF* (*d, t*) / *n* (*t*) and use these terms to find relevance of other documents.
- **Relevance feedback**: Similarity can be used to refine answer set to keyword query
  - User selects a few relevant documents from those retrieved by keyword query, and system finds other documents similar to these
- Vector space model: define an *n*-dimensional space, where *n* is the number of words in the document set.
  - Vector for document *d* goes from origin to a point whose i^{th} coordinate is *TF* (*d,t*) / *n* (*t*)
  - The cosine of the angle between the vectors of two documents is used as a measure of their similarity.


Similarity with threshold Retrieval
Query
{ }
Vector DB
Top k
0.86
0.79
0.77
0.69
0.24
Similarity threshold > 0.75

Relevance Using Hyperlinks

- Number of documents relevant to a query can be enormous if only term frequencies are taken into account
- Using term frequencies makes “spamming” easy
  - E.g., a travel agency can add many occurrences of the words “travel” to its page to make its rank very high
- Most of the time people are looking for pages from popular sites
- Idea: use popularity of Web site (e.g., how many people visit it) to rank site pages that match given keywords
- Problem: hard to find actual popularity of site
  - Solution: next slide

Relevance Using Hyperlinks (Cont.)

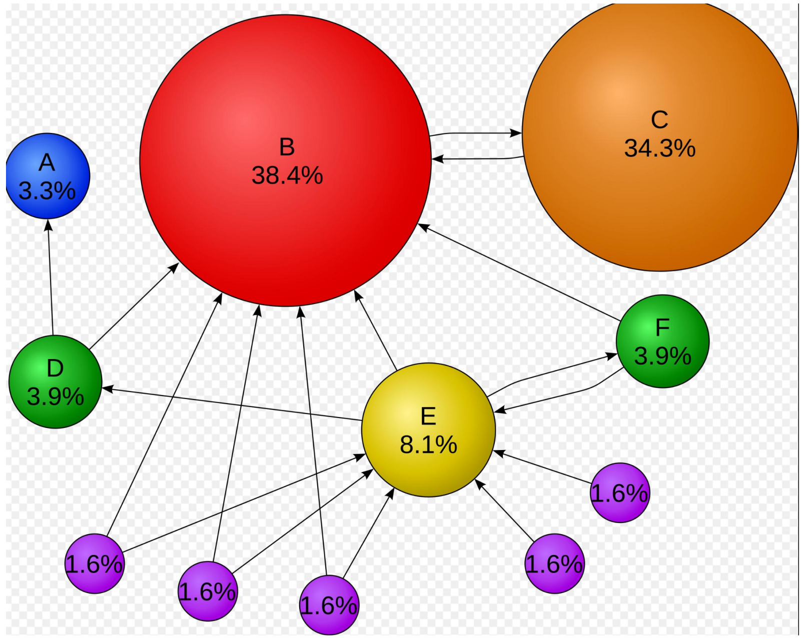
- Solution: use number of hyperlinks to a site as a measure of the popularity or **prestige** of the site
  - Count only one hyperlink from each site (why? - see previous slide)
  - Popularity measure is for site, not for individual page
    - But, most hyperlinks are to root of site
    - Also, concept of “site” difficult to define since a URL prefix like cs.yale.edu contains many unrelated pages of varying popularity
- Refinements
  - When computing prestige based on links to a site, give more weight to links from sites that themselves have higher prestige
    - Definition is circular
    - Set up and solve system of simultaneous linear equations
  - Above idea is basis of the Google **PageRank** ranking mechanism

Relevance Using Hyperlinks (Cont.)

- Connections to **social networking** theories that ranked prestige of people
  - E.g., the president of the U.S.A has a high prestige since many people know him
  - Someone known by multiple prestigious people has high prestige
- Hub and authority based ranking
  - A **hub** is a page that stores links to many pages (on a topic)
  - An **authority** is a page that contains actual information on a topic
  - Each page gets a **hub prestige** based on prestige of authorities that it points to
  - Each page gets an **authority prestige** based on prestige of hubs that point to it
  - Again, prestige definitions are cyclic, and can be got by solving linear equations
  - Use authority prestige when ranking answers to a query

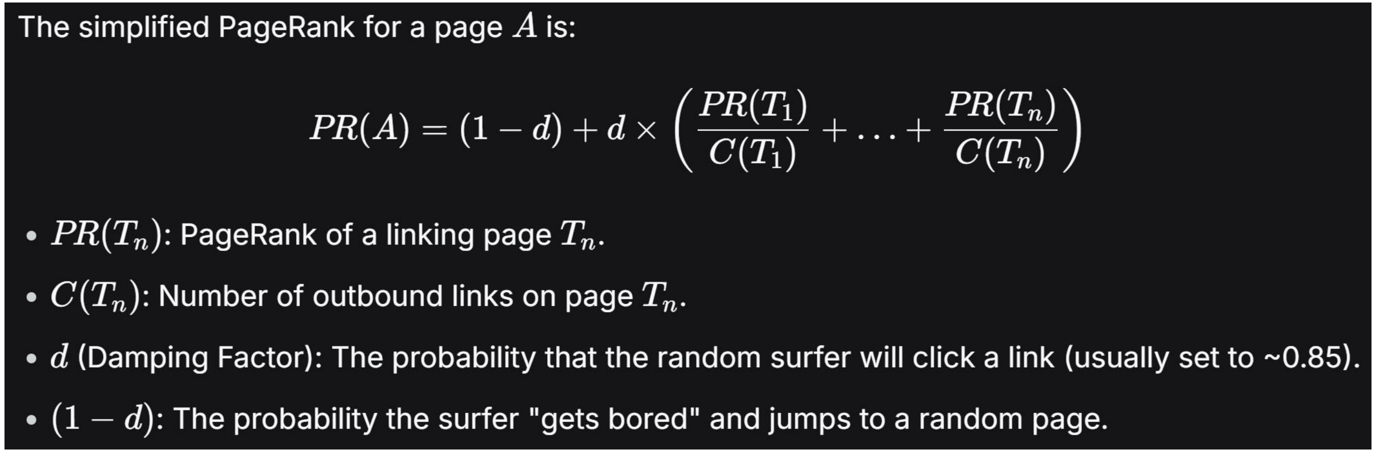
Introduction to PageRank

- What is PageRank?
  - A link analysis algorithm developed by Google founders Larry Page and Sergey Brin at Stanford.
  - Core Idea: A web page is important if it is linked to by other important pages.
  - It assigns a numerical weight to each element of a hyperlinked set of documents (like the World Wide Web) to measure its relative importance.
  - Originally the foundation of Google's web search ranking.
- The Intuition: The "Random Surfer" Model
  - Imagine a person randomly clicking on links.
  - The PageRank value represents the probability that this random surfer will land on a given page.
  - A page has a high rank if:
    - Many pages link to it.
    - Important pages link to it.



How PageRank Works - The Math

- The Basic Formula:



The simplified PageRank for a page A is:

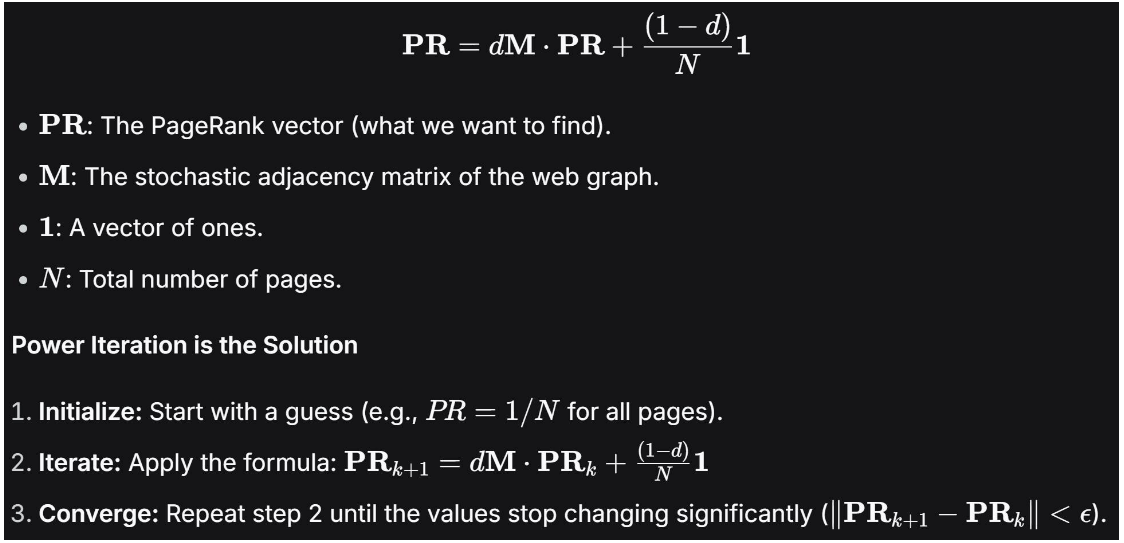
$$PR(A) = (1-d) + d \times \left(\frac{PR(T_1)}{C(T_1)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right)$$

- $PR(T_n)$: PageRank of a linking page T_n.
- $C(T_n)$: Number of outbound links on page T_n.
- d (Damping Factor): The probability that the random surfer will click a link (usually set to ~0.85).
- $(1-d)$: The probability the surfer "gets bored" and jumps to a random page.

- It's a "Chicken and Egg" Problem:
  - To know A's rank, you need to know the rank of all pages linking to A.
  - This creates a system of equations that must be solved simultaneously.

The Power Iteration Connection*

- The entire web's PageRank can be represented as an eigenvector problem.



$$\mathbf{PR} = d\mathbf{M} \cdot \mathbf{PR} + \frac{(1-d)}{N}\mathbf{1}$$

- $\mathbf{PR}$: The PageRank vector (what we want to find).
- $\mathbf{M}$: The stochastic adjacency matrix of the web graph.
- $\mathbf{1}$: A vector of ones.
- N: Total number of pages.

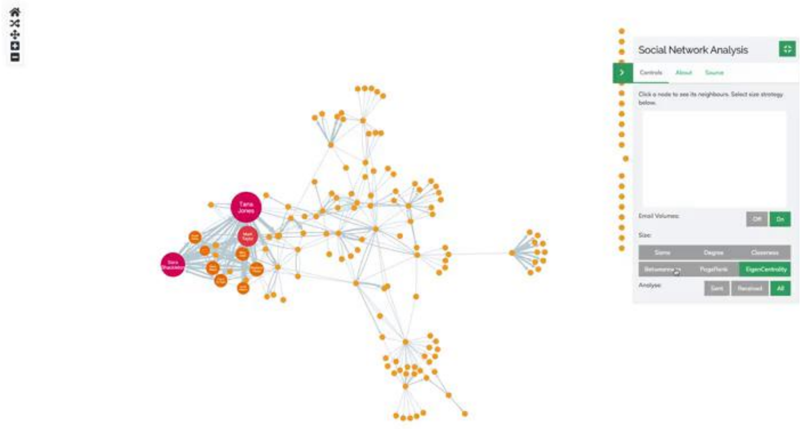
Power Iteration is the Solution

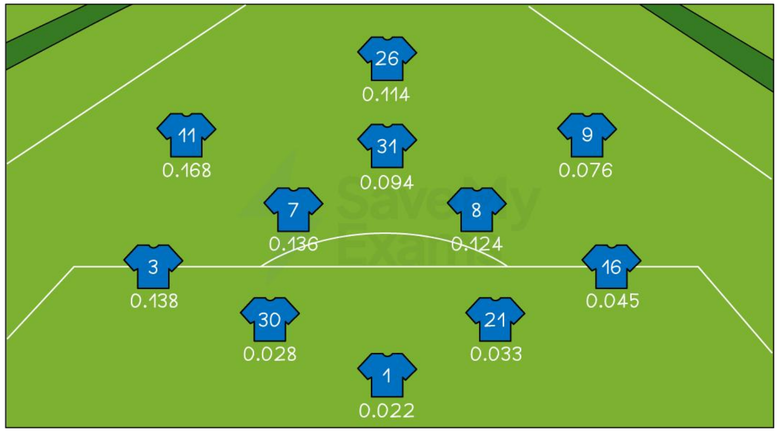
1. **Initialize:** Start with a guess (e.g., $PR = 1/N$ for all pages).
2. **Iterate:** Apply the formula: $\mathbf{PR}_{k+1} = d\mathbf{M} \cdot \mathbf{PR}_k + \frac{(1-d)}{N}\mathbf{1}$
3. **Converge:** Repeat step 2 until the values stop changing significantly ($\|\mathbf{PR}_{k+1} - \mathbf{PR}_k\| < \epsilon$).

- Why it Works:
  - Power Iteration is a numerical method for finding the dominant eigenvector of a matrix.
  - The PageRank vector is the dominant eigenvector of the modified web matrix.
  - This makes the computation scalable to billions of web pages.

Beyond Web Search - Applications

- Social Network Analysis
  - Identifying Influencers: PageRank can identify key influencers in a social graph. A user is "important" if they are followed by other important users.
  - Recommendation Systems: It can rank users or content. For example, on Twitter/X, a user's "Who to Follow" suggestions can be based on a PageRank-like score.
  - Fraud Detection: In a network of financial transactions, PageRank can help identify accounts that are central to suspicious activity.
- Other Domain Applications
  - Bioinformatics: Ranking proteins in a protein-protein interaction network to find the most critical ones.
  - Neuroscience: Identifying key regions in a brain connectivity network.
  - Citation Analysis: Determining the importance of academic papers based on which other papers cite them (the original "Impact Factor" idea).
- Any System that can be Modeled as a Graph: Where the importance of a node depends on the importance of its neighbors.





The Inverted Index: Powering Modern Search

- What is the Problem?
  - We have a massive collection of documents (web pages, articles, books).
  - A user queries for a specific word or phrase.
  - How do we quickly find all documents containing that term without scanning every single document?
- The Answer: The Inverted Index
  - A data structure that maps words to the list of documents they appear in.
  - Think of it as the "index" at the back of a book, but for an entire corpus.
- Analogy: A Book's Index
  - You don't read the entire book to find a topic.
  - You look up the topic in the index, which points you to the relevant pages.

Structure of an Inverted Index

- A Simple Example:
- We have three documents:
  - Doc1: "the cat sat on the mat"
  - Doc2: "the dog played with the cat"
  - Doc3: "the mat is clean"
- Forward Lookup (Inefficient):

Document	Content
Doc1	the cat sat on the mat
Doc2	the dog played with the cat
Doc3	the mat is clean

- Inverted Index (Efficient):

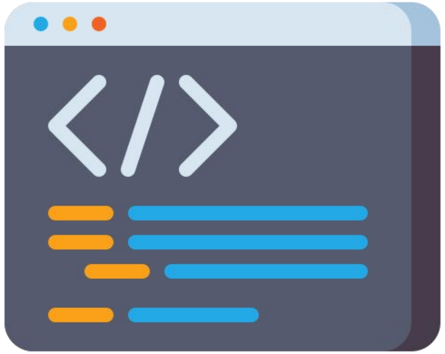
Term	Posting List (Document ID : Positions)
cat	Doc1: [2], Doc2: [6]
dog	Doc2: [3]
mat	Doc1: [6], Doc3: [2]
sat	Doc1: [3]
the	Doc1: [1, 5], Doc2: [1, 5], Doc3: [1]
...	...

- Key Components:
  - Dictionary/Terms: The unique words (e.g., "cat", "mat").
  - Posting List: The list of documents where the term appears.
  - Additional Data (Optional): Term frequency, positions (for phrase queries), etc.

Building an Index with MapReduce

- The Challenge:
  - Building an index for the entire web is impossible on a single machine. We need a distributed approach.
- Why MapReduce?
  - Automatic Parallelization: Splits the work across thousands of machines.
  - Fault Tolerance: Handles machine failures gracefully.
  - Ideal for this task: The process is inherently parallelizable.
- High-Level Process:
  - Input: A large set of raw text documents stored in a distributed file system (e.g., HDFS).
  - MapReduce Phases:
    - Map: Processes each document and emits (key, value) pairs.
    - Shuffle & Sort: Groups all values (intermediate data) by the same key.
    - Reduce: Aggregates the grouped data to form the final index.

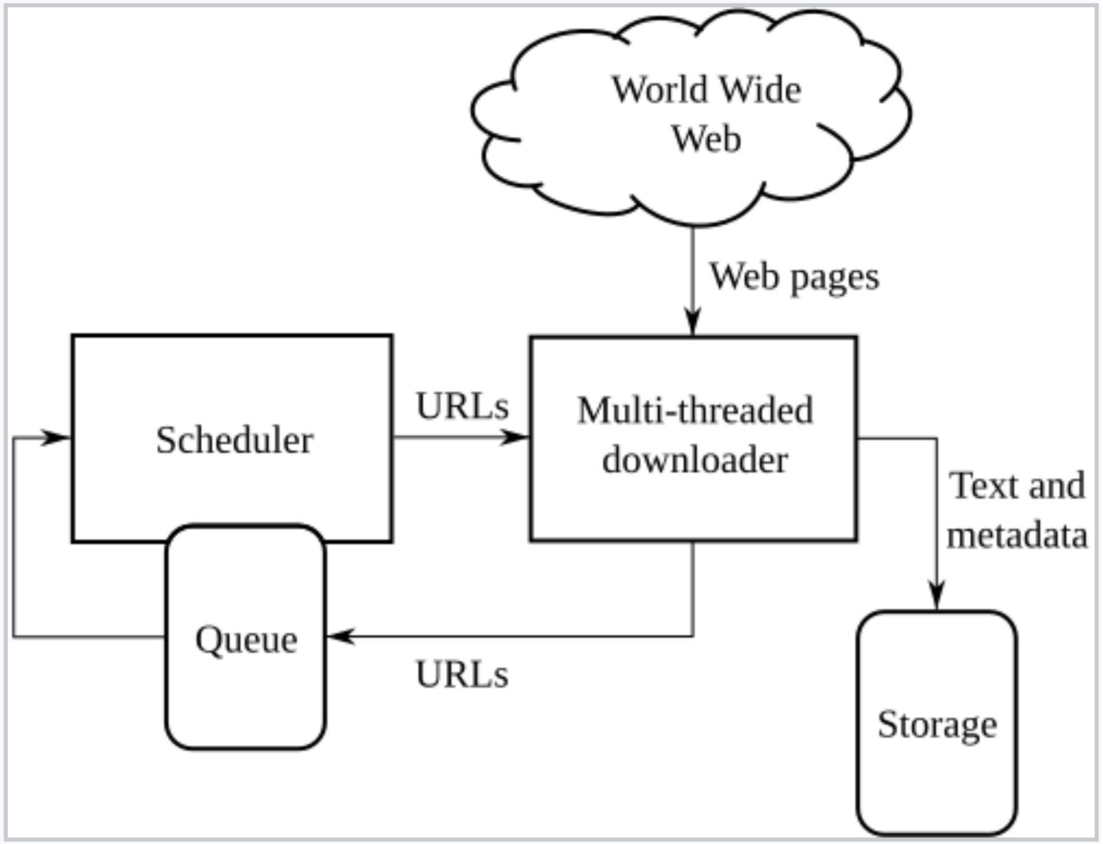
Demo



Web Crawling

- Web crawlers systematically browse the web to collect data.
- Underlying data structures determine efficiency, scalability, and correctness.

Architecture



Example

- https://www.octoparse.com/
- https://www.youtube.com/watch?v=F6CEXNb54TI