Part 1: Background and History of Web Search

1 Introduction to Information Retrieval (IR)

Information retrieval (IR) is the process of obtaining relevant information from a large collection of data (such as documents or web pages) based on user queries. Traditionally, IR dealt with well-structured documents in a controlled environment, but the advent of the World Wide Web introduced a new set of challenges for search engines due to its massive scale, heterogeneity, and dynamic nature.

2 The Birth of the Web and Early Challenges

The web, as we know it today, originated from the concept of hypertext, which was first envisioned by Vannevar Bush in the 1940s and later realized through the development of the World Wide Web in the 1990s. The Web operates on a client-server model where a browser (client) requests data from a web server using a protocol (HTTP - Hypertext Transfer Protocol), and the data is returned in a format (HTML - Hypertext Markup Language) that the browser can render.

Key Concepts:

- **HTTP**: The protocol used to transmit data over the web.
- **HTML**: The markup language used to structure web pages.
- **URL (Uniform Resource Locator)**: Specifies the address of a web resource.

2.1 Web Growth and Scale

The Web's unprecedented growth brought about new challenges for search engines:

- **Unstructured Data**: Unlike traditional information retrieval, web content is created by users with diverse backgrounds and varying levels of expertise.
- **Volume and Diversity**: The Web has billions of pages written in multiple languages and styles.

- **Dynamic Nature**: Pages are often updated frequently, making it difficult for search engines to keep up.

2.2 Early Web Search Engines

Early attempts to make web information discoverable resulted in two types of systems:

- **Full-text Index Search Engines**: Examples include Altavista and Infoseek. These search engines indexed the full text of web pages and allowed users to search using keywords.
- **Taxonomies**: Systems like Yahoo! used a hierarchical structure of categories to organize web pages, allowing users to browse through topics to find relevant information.

2.3 Scale Challenges

Early web search engines had to contend with massive datasets, far larger than any traditional IR system. The number of documents in these indexes often reached tens of millions. To handle this, search engines had to distribute the workload across many machines.

2.4 New Ranking Techniques

Despite the successes of early web search engines, the quality of search results was often poor. The need for new ranking techniques became evident, particularly methods that could measure the **authority** of a document (not just relevance to a query). This led to the development of ranking algorithms that considered links between pages as an indication of authority.

2.5 Hypertext and the Web

The idea of **hypertext**, where text is linked to other text, is central to the Web. Each web page can contain **hyperlinks**, which connect to other web pages, forming a network of information. This decentralized structure makes web search more challenging than traditional IR systems, where documents are often indexed and stored in a centralized database.

2.6 Search Engines and Indexing

Search engines use **crawlers** to gather web pages and create an index of the web. This index contains keywords from each document and is used to quickly match user queries to relevant pages.

Illustrative Example:

Imagine a search engine with 10 million documents. When a user types "best sushi restaurant," the engine searches its index to find documents that contain these keywords. The results are ranked based on relevance and authority.

3 Challenges in Ranking

- **Relevance**: How well does the document match the user's query?
- **Authority**: How trustworthy or influential is the document? This can be measured by looking at how many other web pages link to it.

Numerical Example:

Suppose a search engine uses a simple ranking algorithm where documents are ranked by the number of times a keyword appears in them. If a user searches for "golf clubs" and we have three documents with the following word counts:

- **Document A**: 10 instances of "golf", 5 instances of "clubs"
- **Document B**: 2 instances of "golf", 1 instance of "clubs"
- **Document C**: 7 instances of "golf", 3 instances of "clubs"

The total score for each document can be calculated by summing the occurrences of the keywords:

- **Document A**: $10 + 5 = 15$
- **Document B**: $2 + 1 = 3$
- **Document C**: $7 + 3 = 10$

In this simple ranking scheme, **Document A** would rank first, followed by **Document C** and then **Document B**.

4 Exercise

1. If a search engine indexes 100 million web pages, and each query returns results from 0.01% of the index, how many web pages are returned for each query?
    - **Solution**: 0.01% of 100 million $= 10,000$ pages returned per query.
2. If a user searches for "New York weather" and the top result contains the keywords "New York" 8 times and "weather" 12 times, what is the score for this result if the ranking is based solely on keyword frequency?

- **Solution**: $8 + 12 = 20$ (total score for this result).

In the next part, we'll delve into **Web Characteristics**, **Web Graphs**, and concepts like **spam** detection and prevention in search engines.

5 Web Characteristics, the Web Graph, and Spam

5.1 Web Characteristics

The fundamental characteristics of the Web contribute to the complexities of information retrieval through search engines. The web's **decentralized** and **dynamic** nature means there's no central authority controlling the creation or modification of web content. This poses significant challenges in **content organization**, **indexing**, and **ranking**.

1. **Decentralized Content Creation**: Anyone can create and publish content on the web without coordination or oversight, which leads to:
    - **Diverse languages** and styles of writing.
    - **Variations in content quality**, from professional editorial content to amateur and poorly structured pages.
    - **Heterogeneity**: Differences in layout, design, and content type (text, images, videos, etc.).
2. **Dynamic Content**: Web pages can change frequently, meaning a document indexed today may have different content tomorrow. This includes:
    - **Static Pages**: Content remains consistent over time (e.g., a professor's personal page).
    - **Dynamic Pages**: Content changes based on user interaction or data (e.g., an airport's live flight schedule).
3. **Web Size and Growth**:
    - As of 1995, the size of the web doubled every few months, requiring search engines to scale their systems continuously.
    - Estimating the exact size of the web is difficult due to the presence of **dynamic pages** and varying definitions of what constitutes the "indexable" web.

5.2 The Web Graph

A significant conceptual model of the web is the **web graph**, where each web page is a node, and each hyperlink between web pages is a directed edge.

- **In-links**: Hyperlinks pointing to a web page from other pages.
- **Out-links**: Hyperlinks from a web page to other pages.

Key Characteristics of the Web Graph

- **In-degree and Out-degree**: These metrics represent how many links point to or originate from a page, respectively. For example, if Page A has 10 links pointing to it, its in-degree is 10.
- **Power Law Distribution**: The in-degree distribution of web pages follows a power law, meaning a small number of pages have a very high in-degree (many links pointing to them), while the majority have very few.
  - **Example**: A popular site like Google might have millions of in-links, while a small blog might have only a few.

Bowtie Structure of the Web

The web graph exhibits a **bowtie structure**, which divides the web into three major regions:

- **IN**: Pages from which any page in the **Strongly Connected Component (SCC)** can be reached by following hyperlinks.
- **OUT**: Pages that can be reached from SCC, but there is no way to return to SCC from these pages.
- **SCC (Strongly Connected Component)**: A central portion of the web where every page can reach every other page through hyperlinks.
- **Tendrils and Tubes**: These are pages that are connected to the **IN** or **OUT** regions but do not form part of the core structure.

The **bowtie structure** shows how the web is interconnected, with some regions easily accessible and others isolated.

5.3 Numerical Illustration - Web Graph

Suppose we have a small web with 6 pages (A-F) and the following in-links and out-links:

- **Page A**: In-degree = 1, Out-degree = 1 (links to B)
- **Page B**: In-degree = 3, Out-degree = 1 (links to C)
- **Page C**: In-degree = 1, Out-degree = 1 (links to D)
- **Page D**: In-degree = 1, Out-degree = 1 (links to E)
- **Page E**: In-degree = 1, Out-degree = 0
- **Page F**: In-degree = 0, Out-degree = 0 (isolated page)

In this web graph:

- **Page B** has the highest in-degree (3), indicating it is a central page that others link to.
- **Page F** is an isolated page with no links to or from other pages.

5.4 Spam and Manipulation in Web Search

As web search became a critical tool for users, it also became a target for **spam**—the manipulation of web content to artificially inflate a page's ranking for certain search terms. This is commonly done to drive traffic for commercial gain.

Types of Spam

1. **Keyword Stuffing**: Repeating certain keywords excessively on a page to make it rank higher. For example, a page might include "cheap laptops" hundreds of times in invisible text (e.g., using the same color as the background).
2. **Cloaking:** Showing different content to search engines and users. A search engine crawler might see relevant content, but when a user clicks the link, they are taken to an entirely different, often commercial, page.
3. **Doorway Pages:** Pages that are optimized to rank for specific keywords but then redirect users to another page that may not be relevant.

Combatting Spam

Search engines developed algorithms and techniques to detect and eliminate spam:

- **Invisible text detection**: Search engines now parse HTML for hidden text that users cannot see.
- **Cloaking detection**: They compare the content presented to users with what is indexed by the crawler.
- **Link analysis (explained further in part 3)**: By analyzing the quality and structure of links between pages, search engines can identify manipulated content.

SEO (Search Engine Optimization) and Adversarial Information Retrieval

While legitimate SEO involves optimizing a site for better search rankings (e.g., improving page load time, structuring content well), certain SEO practices aim to deceive search engines. This leads to **adversarial information retrieval**, where search engines and spammers constantly try to outsmart each other.

Numerical Example - Spam Ranking

Consider two pages optimized for the term "buy shoes":

- **Page A** uses keyword stuffing, repeating "buy shoes" 100 times.
- **Page B** uses natural language and includes the keyword "buy shoes" 10 times.

In early search engines, **Page A** might rank higher due to keyword frequency. However, modern algorithms consider user engagement and natural content flow, so **Page B** would likely rank better in contemporary search systems.

5.5 Exercises

1. **Exercise 1**: If a search engine detects that 20% of the web pages in its index are spam, and it indexes 50 million pages, how many spam pages does the search engine index?
   - **Solution**: 20% of 50 million = 10 million spam pages.
2. **Exercise 2**: Suppose a web graph contains 1,000 pages. If the distribution of in-links follows a power law with an exponent of 2.1, what is the probability that a randomly chosen page has exactly 1 in-link?
   - **Solution**: The probability is proportional to $1/i^{2.1}$, where $i = 1$. Therefore, the probability of having 1 in-link is $1/1^{2.1} = 1$.

In the next part, we will explore **advertising models**, the **search user experience**, and methods for **index size estimation** and **duplicate content handling**.

6 Advertising Models, User Experience, and Index Size Estimation

6.1 Advertising as the Economic Model for Web Search

Early on, web search engines realized that advertising could be used to generate revenue. The core idea behind web advertising is to connect users searching for specific information with companies that can meet their needs.

Types of Web Advertising:

1. **Banner Ads**: These are graphical advertisements displayed on popular websites. These ads were usually priced on a **Cost Per Mil (CPM)** basis, meaning advertisers pay for every 1,000 views or impressions of their ad.
2. **Cost Per Click (CPC)**: A more interactive advertising model where advertisers pay only when a user clicks on their ad. The user is then directed to the advertiser's website, where they may purchase a product or service. This model focuses on generating direct actions rather than simply promoting brand awareness.
3. **Search Advertising**: This type of advertising connects users with ads based on the keywords they search for. A user searching for "golf clubs" might see ads for golf clubs alongside their search results.

Sponsored Search

Sponsored search or **search advertising** became a core revenue model for search engines. In this system, advertisers bid for specific keywords, and their ads are displayed when users search for those keywords. Early examples of this model include Goto (later renamed Overture), where advertisers' rankings were based on their bid for a keyword.

Example of Sponsored Search: If three companies bid on the keyword "golf clubs":

- **Company A** bids \$5 per click.
- **Company B** bids \$3 per click.
- **Company C** bids \$1 per click.

Search results for "golf clubs" would show **Company A**'s ad first, followed by **Company B** and **Company C**.

Issues with Sponsored Search:

- **Irrelevant Ads**: Sometimes, the highest bidder may not have the most relevant ad for a query, leading to user dissatisfaction.
- **Click Spam**: Fraudulent clicks on ads by competitors or automated systems can drain an advertiser's budget without generating real leads.

Numerical Example:

If an advertiser pays \$1.50 per click and receives 1,000 clicks on an ad, the total cost would be:

$$\text{Total Cost} = 1.50 \times 1,000 = 1,500 \text{ USD}$$

6.2 The Search User Experience

Search engines need to understand user behavior to improve their services. Web search users differ from traditional IR system users in several ways:

- **Short Queries**: Web users typically enter short, keyword-based queries (often 2-3 words), unlike professional IR users who craft detailed, structured queries.
- **Limited Use of Operators**: Boolean operators (AND, OR, NOT) are rarely used, and users rely on natural language.

User Query Types:

Search queries generally fall into three categories:

1. **Informational Queries**: These seek general information on a topic. For example, "history of the internet" is an informational query where users expect to gather information from multiple pages.
2. **Navigational Queries**: These aim to find a specific website. For example, a user searching for "Amazon" expects to find the homepage of Amazon.com as the first result.
3. **Transactional Queries**: These are intended to perform an action, such as purchasing a product or booking a service. For example, "buy iPhone 15" is a transactional query.

Illustration of User Needs:

Consider a user who searches for "New York Times." This is a **navigational query**, and the user expects the first result to be the official website for The New York Times. Search engines must return highly precise results for navigational queries, aiming for **precision at 1** (the first result being correct).

6.3 Index Size and Estimation

A critical part of a search engine's performance is the size of its index. However, measuring and comparing the index size of different search engines can be difficult due to several factors.

Index Size Challenges:

1. **Dynamic Content**: Many web pages are generated dynamically (e.g., an e-commerce site generates pages based on user input), making it hard to estimate the number of unique pages.
2. **Soft 404s**: Some websites return valid-looking pages (like "Page Not Found" errors) even when no real content exists, further complicating size estimation.

Capture-Recapture Method for Index Size Estimation:

One way to estimate the relative size of two search engines' indexes is through the **capture-recapture method**. This method assumes each search engine indexes a random, independent subset of the web. The goal is to estimate the overlap between the two indexes.

Formulation: Suppose we randomly sample x pages from search engine E_1's index and check if they exist in search engine E_2's index. Similarly, we sample y pages from E_2 and check if they are in E_1.

The relative index sizes of the two engines can be estimated using the formula:

$$\frac{|E_1|}{|E_2|} \approx \frac{y}{x}$$

Where:

- $|E_1|$ is the size of the index for search engine E_1.
- $|E_2|$ is the size of the index for search engine E_2.
- x is the fraction of E_1's pages found in E_2's index.
- y is the fraction of E_2's pages found in E_1's index.

Example:

Let's say:

- 30% of the pages in E_1 are found in E_2's index.
- 50% of E_2's pages are found in E_1's index.

Using the capture-recapture formula:

$$\frac{|E_1|}{|E_2|} = \frac{50}{30} = 1.67$$

Thus, E_1's index is approximately 67% larger than E_2's index.

6.4 Diagrams

Figure 1: Web Graph

This diagram illustrates the **web graph**, showing how web pages are connected via hyperlinks. Each node represents a web page, and the directed edges represent hyperlinks between them.

Figure 2: Bowtie Structure of the Web

This diagram illustrates the **bowtie structure** of the web, consisting of three primary regions:

- **IN**: Pages from which SCC can be reached.
- **SCC (Strongly Connected Component)**: A core set of pages that are all reachable from one another.
- **OUT**: Pages that can be reached from SCC, but are not part of SCC.

Figure 3: Sponsored Search Result Page

This diagram shows the layout of a search results page with **sponsored search results**. The algorithmic results (organic results) appear on the left, while the sponsored search results (paid advertisements) appear on the right.

In the next part, we will delve into handling **near-duplicates** and the process of **shingling** for detecting and removing duplicate or near-duplicate content in search engines. We will also explore methods to address **duplicate detection** and provide further numerical examples.

7 Near-Duplicates, Shingling, and Duplicate Detection

7.1 Handling Near-Duplicates

The web contains a vast amount of duplicated or near-duplicated content, which can take up valuable space in search engine indexes and lead to redundant results being shown to users. Duplicates can arise for various reasons:

- **Mirrored Content**: Some organizations maintain multiple copies of the same web content for backup or load-balancing purposes.
- **Minor Changes**: Web pages might differ only in small details, such as timestamps or advertisements.

7.2 Duplicate Detection via Fingerprinting

One simple approach to detecting duplicates is to use a **fingerprinting** technique. Here, each web page is reduced to a **fingerprint**—a small digest (e.g., 64 bits) of the content. If two web pages have identical fingerprints, they are likely duplicates.

Example: Suppose two pages have the following content:

- **Page A**: "The quick brown fox jumps over the lazy dog."
- **Page B**: "The quick brown fox jumps over the lazy dog."

Using a fingerprinting algorithm, both pages will generate the same fingerprint, indicating they are duplicates. However, this method does not handle **near-duplicates**, which are pages that are almost identical but differ in minor ways, such as different timestamps or advertisements.

7.3 Shingling: Detecting Near-Duplicates

To address the problem of near-duplicates, search engines use a technique called **shingling**. A **shingle** is a sequence of k consecutive terms in a document. The idea is that if two documents share many shingles, they are likely to be near-duplicates.

Example of Shingling:

Consider the sentence: "A rose is a rose is a rose."

- For $k = 4$, the 4-shingles for this text are:
  - "a rose is a"
  - "rose is a rose"
  - "is a rose is"

If two documents share most of their shingles, they are considered near-duplicates.

7.4 Jaccard Coefficient for Measuring Similarity

The **Jaccard coefficient** is used to measure the similarity between two sets of shingles. It is defined as:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Where:

- S_1 and S_2 are the sets of shingles for two documents.
- $|S_1 \cap S_2|$ is the number of shingles that the two documents have in common.
- $|S_1 \cup S_2|$ is the total number of unique shingles in both documents.

Example:

Consider two documents with the following sets of shingles:

- **Document 1**: "a rose is a", "rose is a rose", "is a rose is"
- **Document 2**: "a rose is a", "is a rose is", "rose is beautiful"

The intersection $|S_1 \cap S_2|$ contains 2 shingles: "a rose is a", "is a rose is". ~~The union $|S_1 \cup S_2|$ contains 4 shingles: "a rose is a", "is a rose is", "rose is beautiful", "rose is a rose".~~

The Jaccard coefficient is:

$$J(S_1, S_2) = \frac{2}{4} = 0.5$$

A high Jaccard coefficient (close to 1) indicates the documents are near-duplicates, while a low coefficient indicates they are distinct.

7.5 Hashing for Efficient Duplicate Detection

Calculating the Jaccard coefficient for every pair of documents in a large corpus can be computationally expensive. To reduce this cost, search engines use **hashing** to convert shingles into fixed-size representations. By comparing hashes rather than raw shingles, the process becomes more efficient.

Min-Hashing:

One way to approximate the Jaccard coefficient is to use **Min-Hashing**. This technique works as follows:

- A random permutation is applied to the set of shingles for each document.
- The smallest (minimum) hash value is selected for each document.

- The probability that two documents have the same minimum hash is equal to their Jaccard similarity.

By repeating this process for multiple permutations and keeping track of the matches, an estimate of the Jaccard coefficient can be obtained without directly comparing every pair of shingles.

7.6 Shingle Sketches

A **shingle sketch** is a summary of a document's shingles created using Min-Hashing. If two documents have similar sketches, they are likely near-duplicates.

Example:

Let's say we create sketches for two documents using 200 random permutations. After comparing the sketches, we find that 180 of the minimum hashes match. The estimated Jaccard coefficient would be:

$$J \approx \frac{180}{200} = 0.9$$

This high similarity score suggests the documents are near-duplicates.

7.7 Numerical Example of Near-Duplicate Detection:

Assume two documents have the following shingles:

- **Document A**: 5 unique shingles
- **Document B**: 6 unique shingles
- They share 4 shingles in common.

The Jaccard coefficient can be calculated as:

$$J(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = \frac{4}{7} \approx 0.571$$

This result suggests that the documents are somewhat similar, but not identical.

7.8 Advanced Techniques: Super-Shingles

To further improve efficiency, search engines may use **super-shingles**, which are created by combining several individual shingles into larger units. These super-shingles are then compared across documents, allowing the search engine to quickly eliminate pairs of documents that are unlikely to be duplicates.

7.9 Union-Find Algorithm for Clustering Near-Duplicates

Once potential near-duplicate documents are identified, search engines can group them into **clusters** using the **union-find algorithm**. This algorithm merges documents that share enough similarities into the same cluster, effectively reducing the number of documents that need to be indexed.

7.10 Figures and Diagrams

Figure 1: Shingling Example

This figure shows how a document is broken into shingles. For the text "a rose is a rose is a rose," we extract 4-shingles.

Figure 2: Jaccard Coefficient

This diagram visualizes the Jaccard coefficient between two sets of shingles, showing the intersection and union.

Figure 3: Min-Hashing Process

This diagram illustrates the Min-Hashing process, where random permutations are applied to the shingle sets, and the smallest hash is selected to create a sketch.

Figure 4: Shingle Sketches Comparison

This figure demonstrates how sketches for two documents are compared. Documents with many matching minimum hashes are likely near-duplicates.