html

Sure! Here’s a simplified and detailed restatement of the content:

The Anatomy of a Large-Scale Hypertextual Web Search Engine Sergey Brin and Lawrence Page Stanford University

Abstract

This paper introduces Google , a prototype for a large-scale search engine that leverages the hypertext structure of the web. Google efficiently crawls and indexes the web, delivering superior search results compared to existing systems. The prototype contains a database with over 24 million pages and can be accessed at http ://google .stanford .edu/ .

Key Challenges

Building a search engine is complex. It needs to index millions of web pages and handle vast numbers of unique terms while processing tens of millions of daily queries. Despite their significance, there has been limited academic research on large-scale search engines. The rapid evolution of technology makes creating a search engine today vastly different from three years ago. This paper offers the first detailed public overview of our large-scale search engine.

Key Concepts

- **Scaling Challenges** : Traditional search methods struggle to handle large data volumes.
- **Hypertext Utilization** : New technical challenges arise from leveraging hypertext for improved search results.
- **Uncontrolled Collections** : There’s a need to manage hypertext environments where anyone can publish content.

Keywords

- **World Wide Web**
- **Search Engines**
- **Information Retrieval**
- **PageRank**
- **Google**

1. Introduction

The web poses new challenges for information retrieval due to its rapid growth and the influx of inexperienced users. Many users rely on the web's link graph , starting with curated indices like Yahoo! or automated search engines. While

human-maintained lists efficiently cover popular topics, they are subjective, costly, and slow to update, failing to encompass niche subjects.

Automated search engines often yield numerous low-quality results, exacerbated by advertisers who try to manipulate these systems. Our solution, Google, addresses these challenges by effectively using the additional structure in hypertext, resulting in significantly improved search quality. The name "Google" is a play on googol (10^{100}), reflecting our ambition to create a comprehensive search engine.

Let me know if you need further detail

---

html

Here’s a simplified and detailed restatement of the content:

1.1 Web Search Engines — Scaling Up: 1994 - 2000

Search Engine Technology Growth : Search engines have rapidly scaled to keep pace with the web's expansion. In 1994, the World Wide Web Worm (WWWW) indexed only 110,000 web pages . By November 1997 , leading search engines claimed to index between 2 million (WebCrawler) and 100 million documents. Predictions for the year 2000 suggested a comprehensive index could exceed 1 billion documents.

Query Volume Increase : The number of daily queries has also skyrocketed. For example, the WWWW handled about 1,500 queries per day in early 1994. By late 1997, Altavista managed around 20 million queries daily . With more users and automated queries, major search engines were expected to handle hundreds of millions of queries daily by 2000. The aim of our system is to tackle the quality and scalability issues brought on by this growth.

1.2 Google: Scaling with the Web

Challenges of Scaling : Building a search engine that can scale with the modern web is challenging. Key requirements include:

- **Fast Crawling** : Efficiently gathering and updating web documents.
- **Storage Efficiency** : Effectively storing indices and documents.
- **High-Volume Indexing** : Processing hundreds of **gigabytes** of data.
- **Rapid Query Handling** : Managing hundreds to thousands of queries per second.

Technological Improvements : Although hardware performance and cost have improved, challenges remain, such as disk seek time and operating system robustness . Google is designed to scale with large datasets, using efficient

storage and optimized data structures for quick access. We anticipate that costs related to indexing and storing web content will decrease, favoring centralized systems like Google.

1.3 Design Goals

1.3.1 Improved Search Quality

Main Objective : The primary goal is to enhance the quality of search results. In 1994, there was optimism that a complete search index would make finding information straightforward. However, by 1997 , it became clear that having a comprehensive index is not enough. Users often encounter “junk results” , which dilute relevant results.

User Behavior : As of November 1997, only one of the top four commercial search engines could find its own homepage in the top ten results. The challenge lies in the rapid increase of documents in indices compared to the limited number of results users are willing to review, typically just the first few dozen.

Let me know if you need anything else!

---

html

Here’s a simplified and detailed restatement of the content:

Precision in Search Results

As the size of web collections increases, we require tools that ensure high precision in search results. This means returning only the most relevant documents , particularly in the top ten results. With potentially tens of thousands of slightly relevant documents, achieving this high precision is crucial, even if it sacrifices recall (the total number of relevant documents retrieved).

Recent studies suggest that utilizing more hypertextual information can enhance search performance. Specifically, the link structure and anchor text provide valuable insights for assessing relevance and filtering quality. Google incorporates both of these elements (see Sections 2.1 and 2.2).

1.3.2 Academic Search Engine Research

Over time, the web has shifted from being primarily academic to increasingly commercial . In 1993, only 1.5% of web servers were on .com domains; by 1997, this number rose to over 60% . As a result, search engine development has transitioned from academic settings to commercial enterprises. Most advancements in search technology occur behind

closed doors, resulting in a lack of transparency and a focus on advertising (see Appendix A).

Google aims to bring more development and understanding back into the academic realm . An essential design goal is to create systems that are user-friendly. High usage is vital, as leveraging vast amounts of usage data from modern web systems could lead to significant research insights. However, obtaining this data is challenging because it is considered commercially valuable.

Another design objective is to create an architecture that supports novel research on large-scale web data. Google stores all crawled documents in compressed form, facilitating researchers to process large portions of the web and generate meaningful results. Since its launch, Google has already supported several research papers, with many more in progress. Our goal is to establish an environment where researchers and students can propose and conduct experiments on this extensive data.

2. System Features

The Google search engine boasts two critical features that enhance precision in results:

1. **Link Structure Utilization** : Google calculates a **quality ranking** for each webpage based on the link structure of the web, known as **PageRank** (detailed in [Page 98]).
2. **Link Improvement** : Google leverages links to refine search results further.

2.1 PageRank: Bringing Order to the Web

The web’s citation (link) graph is a significant resource that many existing search engines have underutilized. Google has created maps of up to 518 million hyperlinks , which represent a substantial sample of the total web links. These maps enable the rapid calculation of a webpage’s PageRank , helping to organize and rank web content effectively.

Let me know if you need more information or further clarification!

html

Here’s a simplified and detailed restatement of the content:

PageRank: Measuring Citation Importance

PageRank is an objective measure of a webpage's citation importance, aligning well with users' subjective views of

relevance. This makes PageRank an effective tool for prioritizing search results in response to web keyword searches. For popular topics, a straightforward text match using webpage titles works well when combined with PageRank. In the main Google system, PageRank also significantly enhances full-text searches .

2.1.1 Description of PageRank Calculation

PageRank builds on concepts from academic citation literature by counting backlinks to assess a page's importance or quality. However, PageRank does not treat all links equally; it normalizes the importance based on the number of links on the referring page.

Calculation Formula : For a page A with pages T1 to Tn linking to it, PageRank is calculated as follows:

- **d** : Damping factor (usually set to **0.85**).
- **C(A)** : Number of outgoing links from page A.

The formula is: $PR(A) = (1 - d) + d\left(\frac{PR(T1)}{C(T1)} + \ldots + \frac{PR(Tn)}{C(Tn)}\right)$ $PR(A) = (1 - d) + d(C(T1)PR(T1) + \ldots + C(Tn)PR(Tn))$

Key Points :

- PageRanks form a **probability distribution** over all webpages, meaning their total sums to one.
- PageRank can be computed using an **iterative algorithm** , representing the principal eigenvector of the normalized link matrix.
- Calculating PageRank for **26 million webpages** can be done in a few hours on a medium-sized workstation.

2.1.2 Intuitive Justification

PageRank models user behavior through the concept of a "random surfer." This surfer starts on a random page and clicks on links without going back. The probability of visiting any page reflects its PageRank . The damping factor d represents the likelihood that the surfer will become bored and start on another random page.

Personalization : By applying the damping factor selectively to certain pages, PageRank can incorporate personalization, making it harder to manipulate rankings artificially.

Understanding High PageRank :

- A page earns a high PageRank if:
  - It receives many links from various pages.
  - It has links from high PageRank pages (e.g., a link from **Yahoo!** 's homepage indicates value).

This system recognizes that well-cited pages are generally worth examining. If a page is of poor quality or broken, it's unlikely that reputable pages would link to it. PageRank effectively evaluates these scenarios by recursively spreading

weights through the web’s link structure.

Let me know if

Here’s a simplified and detailed restatement of the content:

2.2 Anchor Text

Anchor Text refers to the clickable text within hyperlinks, and it is treated uniquely in Google’s search engine. Unlike many search engines that only link the anchor text to the page it's on, Google also connects it to the page the link points to. This approach has several benefits:

1. **Accurate Descriptions** : Anchor text often provides more precise descriptions of web pages than the content of those pages.
2. **Indexing Non-Text Content** : Anchor text can be associated with documents that aren’t text-based, such as images, programs, or databases. This allows Google to return results for pages that haven't been crawled, although this can lead to problems if the linked pages don’t exist.

Challenges : While Google sorts results to minimize issues with nonexistent pages, using anchor text efficiently is technically complex due to the massive data involved. For instance, in its current crawl of 24 million pages , Google indexed over 259 million anchors .

2.3 Other Features

Beyond PageRank and anchor text, Google includes several additional features:

- **Location Information** : Google utilizes the location of search hits and emphasizes **proximity** in search results.
- **Visual Presentation** : It considers visual details, such as font size. Words in larger or bolder fonts are given more weight than those in smaller fonts.
- **HTML Repository** : The full raw HTML of pages is stored in a repository for comprehensive access.

3 Related Work

Research on web search engines is relatively brief. One of the earliest search engines was the World Wide Web Worm (WWWW) , developed in 1994. Since then, various academic search engines have emerged, many evolving into public companies. Despite the web's growth and the significance of search engines, there is limited documentation on recent advancements in this area.

According to Michael Mauldin , chief scientist at Lycos Inc., many search services, including Lycos, closely guard their database details. While there is a fair amount of work focused on specific features of search engines, much of this involves post-processing results from existing commercial search engines or creating small-scale personalized search engines.

3.1 Information Retrieval

Research in information retrieval has a long history but typically focuses on small, well-controlled collections, like scientific papers or related news articles. The primary benchmark for this research, the Text Retrieval Conference (TREC) , uses smaller, controlled datasets for evaluation. There is a need for research to adapt these methods for better performance with the vast and varied content available on the web.

Let me know if you need any more details or clarifications!

---

html

Here’s a simplified and detailed restatement of the content:

3.1 Information Retrieval Challenges

The Text Retrieval Conference (TREC) uses a benchmark of only 20GB , while Google's crawl of 24 million web pages spans 147GB . Techniques that work well in TREC often fail on the web. For instance, the vector space model , which compares query and document vectors based on word occurrences, tends to return very short documents that barely address the query. An example includes a major search engine returning a page that simply states "Bill Clinton Sucks" when querying "Bill Clinton."

Some suggest that users should make their queries more specific, but Google strongly disagrees. Users should receive relevant results based on straightforward queries because there is a wealth of high-quality information available. Therefore, traditional information retrieval methods must evolve to effectively handle web content.

3.2 Differences Between the Web and Controlled Collections

The web comprises an uncontrolled and heterogeneous collection of documents, presenting significant variations:

- **Internal Variation** :
  - **Language** : Different human and programming languages.
  - **Vocabulary** : Includes email addresses, links, and product numbers.
  - **Type/Format** : Documents may be text, HTML, PDF, images, or even machine-generated outputs like log files.
- **External Meta Information** :
  - Information inferred about a document but not included within it, such as:
    - **Source Reputation** : Trustworthiness of the content provider.
    - **Update Frequency** : How often the content is refreshed.
    - **Quality** : Overall standard of the content.
    - **Popularity** : Number of views or interactions.

These meta information sources vary greatly; for example, a major site like Yahoo! may receive millions of views daily, while an obscure article might get one view every decade, requiring different handling by search engines.

Key Issues on the Web:

- **Lack of Control** : Anyone can publish content, leading to potential manipulation by companies looking to profit from search engine traffic. This issue is not present in traditional information retrieval systems.
- **Metadata Manipulation** : Attempts to use metadata for better search results often fail because text not visible to users is frequently abused for search engine manipulation. Many companies specialize in exploiting search engines for profit.

4 System Anatomy

The section will cover:

1. **High-Level Architecture** : An overview of the system's design.
2. **Important Data Structures** : Detailed descriptions of key components.
3. **Major Applications** : In-depth examination of the core functions including **crawling** , **indexing** , and **searching** .

Let me know if you need more information or specific details!

html

Here's a simplified and detailed restatement of the content:

4.1 Google Architecture Overview

This section provides a high-level overview of Google's system architecture, which is primarily implemented in C or C++ for efficiency. The system can run on Solaris or Linux .

Key Components of Google's Architecture:

1. **Web Crawling** :
    - Google uses **distributed crawlers** to download web pages.
    - A **URL server** manages and sends lists of URLs to the crawlers.
    - Fetched web pages are sent to a **store server** .
2. **Storing Web Pages** :
    - The store server **compresses** and **stores** web pages in a repository.
    - Each page is assigned a unique identifier known as **docID** when a new URL is processed.
3. **Indexing Process** :
    - The **indexer** reads from the repository, **uncompresses** documents, and parses them into **word occurrences** called **hits** .
    - Each hit records:
        - The **word**
        - Its **position** in the document
        - An **approximation of font size**
        - **Capitalization**
4. **Link Processing** :
    - The indexer also extracts links from web pages and saves this data in an **anchors file** . This file includes:
        - The origin and destination of each link
        - The **text of the link**
    - The **URL resolver** reads the anchors file, converting relative URLs into **absolute URLs** and then to **docIDs** . It associates anchor text with the appropriate docID.
5. **Links Database** :
    - A database of links is created, consisting of pairs of docIDs, which is essential for calculating **PageRanks** .
6. **Sorting and Inverted Index** :
    - The **sorter** rearranges the barrels (initially sorted by docID) into an **inverted index** based on **wordID** . This operation requires minimal temporary space.
    - A program called **DumpLexicon** generates a new lexicon for the searcher, using the wordID list and the lexicon produced by the indexer.
7. **Searching** :
    - The **searcher** runs on a web server, utilizing the lexicon, inverted index, and PageRanks to efficiently respond to user queries.

4.2 Major Data Structures

Google's data structures are designed for efficiency in crawling, indexing, and searching vast document collections. Despite improvements in CPU and I/O rates, disk seeks still take about 10 ms . To minimize disk access, Google's architecture is optimized accordingly, influencing the design of its data structures significantly.

Let me know if you need any further clarification or details!

---

html

Here’s a clearer and detailed restatement of the content:

4.2.1 BigFiles

BigFiles are special virtual files that can span across multiple file systems and are accessible using 64-bit integers . Key features include:

- **Automatic Allocation** : The system manages how data is distributed among different file systems.
- **File Descriptor Management** : BigFiles handle the allocation and deallocation of file descriptors since the operating system may not provide enough for Google's needs.
- **Compression Support** : Basic compression options are available to optimize storage.

4.2.2 Repository

The repository is where the complete HTML content of every web page is stored. Key aspects include:

- **Compression Method** : Pages are compressed using **zlib** (following RFC1950). While **bzip** offers better compression (4:1 ratio), zlib is preferred for its speed (3:1 ratio).
- **Storage Structure** : Documents are stored sequentially, each prefixed with:
  - **docID**
  - **Length**
  - **URL**
- **Ease of Access** : The repository doesn't require additional data structures for access, enhancing data consistency and simplifying development. It allows rebuilding of other data structures using just the repository and a file listing crawler errors.

4.2.3 Document Index

The document index keeps track of all documents. It features:

- **Structure** : Implemented as a fixed-width **ISAM** (Index Sequential Access Method) index, sorted by **docID** .
- **Contents of Each Entry** :
  - Current document status
  - Pointer to the repository

    - Document checksum
    - Various statistics
    - If crawled, a pointer to a variable-width file (**docinfo**) containing the document's **URL** and **title** ; otherwise, it points to a **URLlist** containing just the URL.
- **Efficiency** : The design aims for a compact structure that allows fetching a record in one disk seek during searches.
- **URL to docID Conversion** :
    - A separate file maps **URL checksums** to their corresponding **docIDs** , sorted by checksum.
    - To find a docID for a specific URL, the checksum is computed, and a **binary search** is performed on this file.
    - This method allows for batch conversions, crucial for efficiency, especially with large datasets (322 million links).

4.2.4 Lexicon

The lexicon is structured in a way that allows it to fit in memory affordably. Key points include:

- **Memory Efficiency** : The current lexicon can be stored in memory on machines with **256 MB** of RAM.
- **Size** : It contains **14 million words** , although some rare words are excluded.
- **Implementation** : The lexicon consists of two parts:
    - A concatenated list of words, separated by **nulls** .
    - A **hash table** containing pointers to these words for quick access.

This format breaks down complex concepts into digestible sections while emphasizing key terms and ideas. Let me know if you need further information or clarification!

---

html

Here’s a simplified restatement of the content with important details highlighted:

4.2.5 Hit Lists

Hit lists are records of where a specific word appears in a document, including details like position , font , and capitalization . Key points include:

- **Storage Importance** : Hit lists use a significant amount of space in both the **forward** and **inverted indices** , making efficient representation essential.
- **Encoding Options** : Several encoding methods were considered, such as:
    - Simple encoding (a triple of integers)
    - Compact encoding (optimized bit allocation)
    - **Huffman coding**

  Ultimately, a **hand-optimized compact encoding** was chosen for its efficiency in space usage.

Hit List Structure

- **Size** : Each hit occupies **two bytes** .
- **Types of Hits** :
  - **Fancy Hits** : Occur in URLs, titles, anchor text, or meta tags.
  - **Plain Hits** : Include all other occurrences.
- **Plain Hit Composition** :
  - **Capitalization bit** : Indicates if the word is capitalized.
  - **Font size** : Represented using **three bits** (7 possible values; 111 indicates a fancy hit).
  - **Word position** : Stored using **12 bits** (positions above 4095 are labeled as 4096).
- **Fancy Hit Composition** :
  - **Capitalization bit**
  - **Font size** : Set to 7 to signify it's a fancy hit.
  - **Type of fancy hit** : Encoded in **4 bits** .
  - **Position** : Stored in **8 bits** , split into parts for anchor position and a hash of the docID.
- **Phrase Searching** : Limited phrase searching is possible due to how anchor hits are stored. Future updates aim to enhance the resolution of these fields.
- **Hit List Length** : The length of the hit list is recorded before the actual hits.
  - **Combination for Space Saving** : The length is combined with the **wordID** in the forward index and **docID** in the inverted index.
  - If the length exceeds the available bits, an **escape code** signals that the actual length follows in the next two bytes.

4.2.6 Forward Index

The forward index is structured for efficiency:

- **Partially Sorted** : It is initially sorted and organized into **barrels** (64 used).
- **Barrel Functionality** : Each barrel contains:
  - A range of **wordIDs** .
  - The **docID** of documents that contain those words, followed by lists of corresponding wordIDs with hit lists.
- **Storage Efficiency** : Although this method leads to slight duplication of docIDs, the impact is minimal for a reasonable number of barrels. This approach also simplifies the final indexing process done by the sorter.
- **Relative WordID Storage** : Instead of storing actual wordIDs, they are stored as relative differences from the smallest wordID in the barrel, optimizing space further.

This summary emphasizes clarity while maintaining essential details about the architecture and structure of Google's indexing system. If you have more content to go over or need further assistance, just let me know!

Here’s a simplified restatement of the content with important details highlighted:

4.2.7 Inverted Index

The inverted index is similar to the forward index , but it is processed for efficient querying:

- **Barrel Structure** : It uses the same barrels as the forward index. Each barrel corresponds to a **wordID** and includes a pointer in the **lexicon** to a **doclist** containing **docIDs** and their associated hit lists.

Doclist Order

The order of docIDs in the doclist is crucial:

- **Sorted by docID** : This allows for quick merging of different doclists, making it easy to handle multiple word queries.
- **Sorted by Ranking** : While this makes one-word queries faster and likely places relevant results at the top, merging becomes complicated, and any change in ranking requires rebuilding the index.

Compromise Approach : Google uses two sets of inverted barrels:

- One set includes hit lists with title or anchor hits.
- The second set contains all hit lists.

This strategy checks the first set for matches before expanding to the larger set.

4.3 Crawling the Web

Web crawling is a complex task that involves several challenges:

- **Performance and Reliability** : Interacting with numerous web servers introduces variability and potential issues.
- **Social Challenges** : Many people may not understand what a crawler is, leading to inquiries about its activities.

Google’s Crawling System

Google employs a distributed crawling system to handle vast amounts of data:

- **URL Server** : A single **URLserver** distributes URL lists to multiple crawlers (typically around three).
- **Crawler Connections** : Each crawler maintains about **300 open connections** to fetch web pages quickly, achieving speeds of over **100 pages per second** (approximately **600K bytes per second**).

Technical Considerations

- **DNS Lookups** : Each crawler has its own **DNS cache** to avoid delays from repeated lookups.
- **State Management** : Crawlers manage multiple connection states (DNS lookup, connection, request, response) using **asynchronous I/O** and event-driven queues.

Community Interaction

Running a crawler that accesses over half a million servers results in significant community feedback:

- **Common Inquiries** : Many webmasters express curiosity about the crawler's activities, often unaware of the **robots exclusion protocol** .
- **Unexpected Issues** : Crawling certain sites (like online games) can lead to unintended disruptions, but most issues are manageable.

This summary aims to clarify the concepts and processes involved in the inverted index and web crawling while retaining key details. Let me know if you need more information or have other sections to cover!

---

html

Here's a simplified and detailed restatement of the content, with key ideas and terms bolded:

4.4 Indexing the Web

Challenges of Crawling

- **Testing a Crawler** : It’s challenging to test a web crawler because it must handle a vast range of web pages and servers. Problems can arise from obscure issues on individual pages, potentially causing crashes or unpredictable behavior.
- **Robust Design** : Crawlers need to be designed to be **robust** and thoroughly tested. Given their complexity, significant resources must be dedicated to addressing issues as they arise, including monitoring user feedback

and emails.

Parsing Documents

- **Error Handling** : A web parser must manage various errors, such as:
  - Typos in HTML tags
  - Invalid characters or deeply nested tags
  - Unexpected formats that may confuse standard parsers.
- **Parser Design** : For speed and robustness, Google uses **flex** to create a **lexical analyzer** with its own stack, instead of using a traditional **CFG parser** like YACC. This approach required significant development effort to ensure the parser is both efficient and reliable.

Indexing Documents into Barrels

- **Encoding Process** : After parsing, documents are encoded into barrels:
  - Each word is assigned a **wordID** using an **in-memory hash table** called the **lexicon** .
  - New words not in the base lexicon (which contains 14 million words) are logged separately to allow for parallel processing of indexing.
- **Hit Lists** : Word occurrences in documents are recorded as **hit lists** and stored in the **forward barrels** .

Sorting for Inverted Index

- **Generating Inverted Index** : To create the inverted index, the sorter processes each forward barrel:
  - Sorts by **wordID** to produce separate inverted barrels for title and anchor hits, and another for full text.
  - This sorting is done one barrel at a time, minimizing the need for temporary storage.
- **Parallel Sorting** : The sorting process is parallelized by using multiple machines to handle different barrels simultaneously. If barrels are too large for main memory, they are subdivided into smaller **baskets** that fit into memory, sorted, and then written into the inverted barrels.

4.5 Searching

- **Search Efficiency and Quality** : The goal of the search process is to deliver **high-quality results efficiently** . While many commercial search engines focus on improving efficiency, Google emphasizes enhancing the quality of search results.
- **Scalability** : Google believes its solutions can scale to commercial levels with further optimization.
- **Query Evaluation** : The overall query evaluation process in Google is depicted in **Figure 4** .

This summary aims to clarify the indexing and searching processes involved in web crawling, while retaining essential details. Let me know if you need further assistance or additional sections covered!

Here’s a clearer and detailed restatement of the content, with key ideas and terms bolded :

4.5 Searching and Ranking

Response Time Limitation

- To manage **response time** , Google limits the number of matching documents to **40,000** . Once this limit is reached, the search process skips to step 8 in the query evaluation process, which can lead to **sub-optimal results** . Ongoing investigations are exploring better solutions for this limitation. In the past, sorting hits by **PageRank** improved outcomes.

4.5.1 The Ranking System

- Google collects **extensive information** about web documents, unlike typical search engines. Each **hitlist** includes details like position, font, and capitalization, along with hits from **anchor text** and the document’s **PageRank** .
- **Ranking Function** :
  - The goal is to balance the influence of various factors in ranking documents.
  - For a **single-word query** :
    1. Google checks the hitlist for the queried word in each document.
    2. Hits are categorized by type (e.g., title, anchor, URL) and assigned **type-weights** .
    3. The number of hits of each type is counted, and these counts are transformed into **count-weights** , which increase initially but taper off at higher counts.
    4. The **dot product** of the count-weights vector and the type-weights vector produces an **Information Retrieval (IR) score** .
    5. This score is then combined with the document’s PageRank to yield the final document rank.
- For a **multi-word search** :
  - Multiple hitlists are examined, focusing on the proximity of hits:
    - Nearby hits are given more weight than those further apart.
    - Proximity is measured and classified into **10 value "bins"** , ranging from close matches to distant hits.
    - Counts for each type and proximity are computed, with each pair having a **type-prox-weight** .
    - The dot product of count-weights and type-prox-weights calculates the IR score.
- These computations can be displayed in a **debug mode** , which aids in refining the ranking system.

4.5.2 Feedback Mechanism

- The ranking function involves numerous parameters, such as type-weights and type-prox-weights, whose optimal values are somewhat elusive.
- A **user feedback mechanism** allows trusted users to evaluate search results. This feedback is recorded, enabling Google to assess the impact of changes to the ranking function on previously ranked searches. Although not perfect, this approach aids in refining the ranking system.

Google Query Evaluation Process

1. **Parse the query** .
2. **Convert words into wordIDs** .
3. **Seek the start** of the doclist in the short barrel for each word.
4. **Scan through doclists** until a document matches all search terms.
5. **Compute the document rank** for the query.
6. If at the end of any doclist in the short barrels, **seek to the full barrel** for each word and return to step 4.
7. If not at the end of any doclist, go back to step 4.
8. **Sort matched documents** by rank and return the top results.

This summary simplifies the search and ranking processes while keeping essential details intact. Let me know if you need more information or further clarifications!

---

html

Here’s a simplified and detailed restatement of the content, with key ideas and terms bolded :

---

5 Results and Performance

Quality of Search Results

- The **quality of search results** is the primary measure of a search engine's effectiveness. Although a comprehensive user evaluation isn't feasible in this paper, Google has consistently produced better results than major commercial search engines for most queries.
- For instance, when searching for **"Bill Clinton"** , Google's results utilize **PageRank** , **anchor text** , and **proximity** to provide relevant information.

Features of Google’s Search Results

- The search results are **clustered by server** , which makes it easier for users to navigate through the results.
- Many results come from the **whitehouse.gov** domain, which is expected for such a search. In contrast, most commercial search engines do not return results from this domain, let alone the correct ones.
- The first result lacks a title because it was not directly crawled; instead, Google used **anchor text** to identify it as a relevant answer. The fifth result, which is an email address, is also determined by anchor text, as it is not crawlable.
- All displayed results are high-quality pages, and none were found to be broken links, primarily due to their high **PageRank** . The PageRank percentages are shown in red alongside **bar graphs** for visual representation.

- The search specifically focuses on **Bill Clinton** , with no irrelevant results, thanks to the importance placed on the **proximity** of word occurrences in the documents.
- A thorough evaluation of a search engine's quality typically requires extensive user studies, which is beyond this document's scope. Readers are encouraged to try Google themselves at http ://google .stanford .edu .

5.1 Storage Requirements

- Example query: **bill clinton** results include various links from **whitehouse.gov** and other related sites, showcasing their **PageRank** scores.

This summary highlights the essential points about the effectiveness and features of Google's search results while retaining key details. If you need further clarification or additional information, let me know!

---

html

Here's a clearer restatement of the content with detailed notes and key terms bolded for emphasis:

Storage and System Performance

5.1 Storage Efficiency

- Google is designed to scale **cost-effectively** as the web grows. Efficient **storage** is a key aspect of this.
- According to **Table 1** , the total size of Google's data repository is about **53 GB** , which is only one-third of the total data stored.
- Given current disk prices, this makes the repository a **relatively cheap** source of useful data.
- The total storage needed for all data used by the search engine is approximately **55 GB** . Most queries can be answered using just the **short inverted index** .
- With further improvements in encoding and compression, a high-quality web search engine could fit on a **7 GB drive** of a new PC.

5.2 System Performance

- Efficient **crawling** and **indexing** are crucial for keeping information updated and for testing system changes quickly.
- Major operations for Google include **Crawling** , **Indexing** , and **Sorting** .
- Measuring the total time for crawling is challenging due to various interruptions (e.g., filled disks, server crashes).
- Overall, it took about **9 days** to download **26 million pages** (including errors). Once stabilized, the last **11 million pages** were downloaded in just **63 hours** , averaging over **4 million pages per day** or **48.5 pages per second** .

- The **indexer** and **crawler** operated simultaneously, with the indexer optimized to avoid being a bottleneck, running at about **54 pages per second** .
- **Sorting** can be done in parallel; using four machines, the sorting process takes around **24 hours** .

5.3 Search Performance

- While improving **search performance** hasn't been the main focus, the current version of Google typically answers queries within **1 to 10 seconds** .
- This response time is primarily affected by **disk I/O** over NFS (Network File System), as the data is spread across multiple machines.
- Currently, Google lacks optimizations such as **query caching** and **subindices** for common terms, but plans are in place to enhance speed through **distribution** , **hardware** , **software** , and **algorithmic** improvements. The target is to handle several hundred queries per second.
- **Table 2** presents sample query times from the current version, showing speed improvements resulting from cached I/O.

Storage Statistics (Table 1 Overview)

- **Total Size of Fetched Pages** : **147.8 GB**
- **Compressed Repository** : **53.5 GB**
- **Short Inverted Index** : **4.1 GB**
- **Full Inverted Index** : **37.2 GB**
- **Lexicon** : **293 MB**
- **Temporary Anchor Data** : **6.6 GB**
- **Document Index (including variable width data)** : **9.7 GB**
- **Links Database** : **3.9 GB**
- **Total Without Repository** : **55.2 GB**
- **Total With Repository** : **108.7 GB**

Web Page Statistics

- **Number of Web Pages Fetched** : **24 million**
- **Number of URLs Seen** : **76.5 million**
- **Number of Email Addresses** : **1.7 million**
- **Number of 404 Errors** : **1.6 million**

This version emphasizes the essential details while simplifying the explanation for better understanding. If you have any specific sections you want to delve into more or need further clarification, just let me know!

Here’s a clearer and more concise restatement of the content with key terms bolded for emphasis:

6. Conclusions

Google is designed as a scalable search engine focused on providing high-quality search results across the rapidly expanding World Wide Web. To enhance search quality, Google uses several techniques, including PageRank , anchor text , and proximity information . It features a comprehensive architecture for gathering web pages , indexing them, and conducting search queries .

6.1 Future Work

Creating a large-scale web search engine is complex, and there is much left to accomplish. Immediate goals include:

- **Improving search efficiency** and scaling to about **100 million web pages** .
- Simple enhancements such as **query caching** , **smart disk allocation** , and **subindices** .
- Developing effective algorithms to determine which old pages need to be recrawled and which new pages should be crawled. Previous research in this area includes work by **Cho 98** .
- Exploring the use of **proxy caches** to build demand-driven search databases.

Future plans also include adding features commonly supported by commercial search engines, like:

- **Boolean operators**
- **Negation**
- **Stemming**

Additional areas of exploration involve:

- **Relevance feedback** and **clustering** (currently, Google offers basic hostname-based clustering).
- Supporting **user context** (e.g., user location) and **result summarization** .
- Enhancing the use of **link structure** and **link text** . Preliminary experiments suggest that **PageRank** can be personalized by increasing the weight of a user’s home page or bookmarks. Experiments are also underway to include surrounding text in addition to link text.

The web search engine environment offers a wealth of research opportunities, and many ideas are yet to be explored, indicating that this Future Work section will remain extensive.

6.2 High Quality Search

The primary challenge for users of web search engines is the quality of search results . While search results can be entertaining, they often lead to frustration and wasted time. For example, one popular search engine's top result for "Bill Clinton" was a joke rather than relevant content.

Google aims to deliver higher quality search results as the web expands, making it easier to find information. To achieve

this, Google leverages:

- **Hypertextual information** , including link structure and **anchor text** .
- **Proximity** and **font information** .

Although evaluating a search engine is complex, subjective assessments suggest that Google provides higher quality results compared to existing commercial search engines. The analysis of link structure through PageRank helps assess web page quality, while link text provides descriptive context. Additionally, proximity information significantly enhances relevance for many queries.

Search Times (Table 2 Overview)

Query	CPU Time (s)	Total Time (s)	CPU Time (s)	Total Time (s) (Cached)
al gore	0.09	2.13	0.06	0.06
vice president	1.77	3.84	1.66	1.80
hard disks	0.25	4.86	0.20	0.24
search engines	1.31	9.63	1.16	1.16

This version simplifies the content while maintaining detail and highlighting key concepts. If you need more specific insights or further elaboration on any part, feel free to ask!

html

Here's a simplified and concise restatement of the content, with key terms bolded for emphasis:

6.3 Scalable Architecture

Google is designed to be scalable , ensuring efficiency in both space and time . Managing constant factors is crucial when dealing with the vastness of the Web. Throughout its development, Google has encountered bottlenecks in several areas:

- **CPU** performance
- **Memory access** and **capacity**
- **Disk seeks** and **throughput**
- **Network I/O**

To address these challenges, Google has evolved its systems to overcome these bottlenecks during various operations. Key points include:

- Major data structures are optimized for **efficient storage use** .
- The processes of **crawling** , **indexing** , and **sorting** are designed to quickly build a substantial web index. For instance, Google can index **24 million pages in under a week** and aims to index **100 million pages in less than a month** .

6.4 A Research Tool

Beyond being a high-quality search engine, Google serves as a valuable research tool . The data collected by Google has already led to numerous research papers, with more expected in the future. Research, such as that by Abiteboul 97 , highlights certain query limitations that can be addressed without having a local copy of the Web.

This shows that Google—or similar systems—are essential for various applications, making them beneficial for researchers and searchers globally. Google is anticipated to inspire the next generation of search engine technology .

7. Acknowledgments

The development of Google owes much to Scott Hassan and Alan Steremberg for their critical contributions. Gratitude is also extended to Hector Garcia-Molina , Rajeev Motwani , Jeff Ullman , Terry Winograd , and the WebBase group for their support and insightful discussions. Additionally, appreciation is given to equipment donors like IBM , Intel , and Sun , along with funding support from various organizations including:

- **National Science Foundation**
- **DARPA**
- **NASA**
- **Interval Research**

This research was part of the Stanford Integrated Digital Library Project .

References

- Best of the Web 1994: Navigators
- Bill Clinton Joke of the Day: April 14 , 1997
- Bzip2 Homepage: Bzip2
- Google Search Engine: Google
- Harvest: Harvest
- Lycos Design Choices: IEEE Expert Interview
- Effect of Cellular Phone Use: WebFirst
- Search Engine Watch: Search Engine Watch
- RFC 1950 (zlib): zlib
- Robots Exclusion Protocol: Robots Exclusion

This version maintains detail while ensuring clarity and emphasizing key concepts. If you need further elaboration or

Here's a clearer and concise restatement of the provided content, with key terms bolded for emphasis:

Web Growth Summary

Web Growth refers to the rapid expansion of the internet and its impact on information retrieval and accessibility. A detailed summary can be found at: MIT Web Growth Summary .

Key References

1. **Yahoo!** : A major web directory and search engine, known for its early role in web navigation. Visit: Yahoo .
2. **[Abiteboul 97]** : Serge Abiteboul and Victor Vianu discuss **queries** and **computation** on the Web in their paper presented at the International Conference on Database Theory in Greece, 1997.
3. **[Bagdikian 97]** : Ben H. Bagdikian's book, **The Media Monopoly** , 5th Edition, explores the impact of media ownership on information diversity (ISBN: 0807061557).
4. **[Cho 98]** : Junghoo Cho, Hector Garcia-Molina, and Lawrence Page address **efficient crawling** strategies in their work at the Seventh International Web Conference (WWW 98) in Brisbane, 1998.
5. **[Gravano 94]** : Luis Gravano, Hector Garcia-Molina, and A. Tomasic evaluate the effectiveness of **GlOSS** for text-database discovery at the 1994 ACM SIGMOD International Conference.
6. **[Kleinberg 98]** : Jon Kleinberg's work on **authoritative sources** in hyperlinked environments was presented at the ACM-SIAM Symposium on Discrete Algorithms in 1998.
7. **[Marchiori 97]** : Massimo Marchiori discusses the pursuit of **correct information** on the Web and introduces **hyper search engines** at the Sixth International WWW Conference in 1997.
8. **[McBryan 94]** : Oliver A. McBryan presents tools like **GENVL** and **WWWW** for managing the web at the First International Conference on the World Wide Web in Geneva, 1994.
9. **[Page 98]** : Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd describe the **PageRank citation ranking** in a manuscript, highlighting its role in organizing web information.
10. **[Pinkerton 94]** : Brian Pinkerton shares insights from **WebCrawler** in finding user information at the Second International WWW Conference in Chicago, 1994.
11. **[Spertus 97]** : Ellen Spertus discusses **mining structural information** on the Web in her presentation at the Sixth International WWW Conference.
12. **[TREC 96]** : Proceedings from the **fifth Text REtrieval Conference (TREC-5)** provide valuable insights into text retrieval methodologies. Access the full text at: TREC .
13. **[Witten 94]** : Ian H. Witten, Alistair Moffat, and Timothy C. Bell authored **Managing Gigabytes** , which focuses on **compressing** and **indexing** large volumes of documents and images.
14. **[Weiss 96]** : Ron Weiss et al. present **HyPursuit** , a **hierarchical network search engine** that utilizes content-link hypertext clustering, discussed at the 7th ACM Conference on Hypertext in 1996.

Vitae

This section likely refers to the professional background and credentials of the authors or contributors involved in the aforementioned research and references.

This format organizes the information clearly and highlights essential details for better understanding. If you have any specific areas you'd like to dive deeper into, feel free to ask!

Here’s a clearer and more concise restatement of the content, highlighting key terms and ideas:

Sergey Brin

- **Education** :
  - B.S. in **Mathematics and Computer Science** from the **University of Maryland** , College Park (1993).
  - M.S. in **Computer Science** from **Stanford University** (1995).
  - Currently a **Ph.D. candidate** in Computer Science at Stanford.
- **Achievements** :
  - Recipient of a **National Science Foundation Graduate Fellowship** .
- **Research Interests** :
  - **Search engines** .
  - **Information extraction** from unstructured data.
  - **Data mining** of large text collections and scientific data.

Lawrence Page

- **Education** :
  - B.S.E. in **Computer Engineering** from the **University of Michigan, Ann Arbor** (1995).
  - Currently a **Ph.D. candidate** in Computer Science at Stanford.
- **Research Interests** :
  - **Link structure** of the web.
  - **Human-computer interaction** .
  - **Search engines** .
  - **Scalability** of information access interfaces.
  - **Personal data mining** .

Appendix A: Advertising and Mixed Motives

- **Business Model** :
  - Most commercial search engines primarily rely on **advertising** .
  - This model can conflict with the goal of providing high-quality search results for users.
- **Example of Conflict** :
  - In their prototype search engine, the top result for "cellular phone" was a study on the risks of using phones while driving. This result ranked high due to **PageRank** , which measures **citation importance** on the web.
  - An advertising-funded search engine might struggle to justify showing this result to advertisers focused on promoting phone sales.
- **Bias in Search Engines** :
  - Advertising can lead to **bias** favoring advertisers over users.
  - **Historical Context** : Experiences with media suggest that ad-funded models tend to prioritize advertiser needs.
- **Difficulties in Evaluation** :
  - Evaluating search engine quality is challenging, even for experts.
  - **OpenText** was noted for selling top listing rights, creating significant bias in results.
- **Insidious Bias** :
  - Less obvious biases can exist, such as favoring "friendly" companies while downgrading competitors, which are harder to detect but impactful.
- **Market Tolerance** :
  - The market may tolerate subtle biases, which can still significantly affect outcomes.
  - Advertising income can lead to **poor quality** search results as the focus shifts away from user needs.

This summary organizes the content in a clear manner while emphasizing key concepts and terms for better understanding. Let me know if you need more information or further clarification!

---

html

Here’s a simplified and organized restatement of the content, emphasizing key concepts:

Mixed Incentives in Advertising

- **Advertising Impact** :
  - A major search engine failed to return a **large airline's homepage** when its name was searched because the airline had placed a costly advertisement linked to its name.
  - This situation highlights that a **better search engine** should deliver relevant results without relying on ads, potentially reducing revenue for the search engine.
- **Consumer Perspective** :

- From a **consumer viewpoint** , effective search engines would minimize the need for advertisements to find desired information.
- This poses a challenge to the existing **advertising-supported business model** of many search engines.

- **Market Dynamics** :
  - Despite this, advertisers will always seek to promote product switches or new offerings.
  - It's essential to have a **competitive search engine** that prioritizes transparency and operates within an **academic framework** to address these mixed incentives.

Appendix B: Scalability

9.1 Scalability of Google

- **Design Goals** :
  - Google is designed to be scalable, aiming to index up to **100 million web pages** .
  - New hardware has been acquired to support this capacity, and key processes (like **crawlers** , **indexers** , and **sorters**) can operate in parallel, ensuring efficiency.
- **Operating Limits** :
  - Scaling to 100 million pages will approach **operating system limits** , such as:
    - Addressable memory.
    - Number of open file descriptors.
    - Network sockets and bandwidth.
  - Expanding beyond this limit may significantly increase system complexity.

9.2 Scalability of Centralized Indexing Architectures

- **Text Indexing Potential** :
  - With advancements in computing, indexing vast amounts of text is becoming feasible and cost-effective.
  - As production costs for text are low compared to more demanding media (like video), text will likely remain widely available.
  - Advances in **speech recognition** could further increase the volume of text data.
- **Illustrative Example** :
  - If every person in the US (approximately **250 million**) wrote an average of **10,000 words** daily, it would total around **850 terabytes** of text annually.
  - Assuming reasonable costs for indexing, and linear indexing methods, we can estimate indexing time and costs under these conditions.
- **Moore's Law** :
  - **Moore's Law** states that computing power doubles approximately every **18 months** .
  - If this trend continues, only **10 more doublings** (about **15 years**) would be needed to index all the written content in the US for an affordable price.
  - While some experts are skeptical about the continued validity of Moore's Law, numerous valuable centralized applications can emerge even if only partial progress is made.

This summary captures the essence of the original content while making it more accessible. Let m

Here’s a simplified and organized restatement of the content, focusing on key concepts:

Distributed vs. Centralized Indexing Systems

- **Distributed Systems** :
  - **Gloss** and **Harvest** are examples of distributed systems that can provide efficient and elegant solutions for **indexing** data.
  - However, convincing users to adopt these systems is challenging due to **high administration costs** associated with managing multiple installations.
- **Potential for Cost Reduction** :
  - If administration costs can be significantly reduced, more users may adopt distributed indexing systems.
  - This shift could lead to substantial improvements in **search efficiency** and quality.

Scalability of Text Indexing

- **Human vs. Machine Content** :
  - Humans can only produce a limited amount of content through typing or speaking, while machines can generate content indefinitely.
  - Focusing on indexing **human-generated content** is seen as extremely valuable and practical.
- **Optimistic Outlook** :
  - The authors are optimistic that the **centralized web search engine architecture** will enhance its ability to index relevant text information over time.
  - There is a positive outlook for the future of search capabilities as technology continues to improve.

This summary captures the main ideas in a clear and concise way. Let me know if