Search Engine Project Documentation

1. System Design

1.1 Overview

The search engine system employs a modular architecture comprising two primary components designed to efficiently manage data acquisition, indexing, and retrieval:

* Web Crawler and Indexer (HtmlParser)
* Search Engine (SearchEngine)

The modular approach significantly facilitates independent development, simplified testing processes, enhanced maintainability, and scalability of the entire system.

1.2 Web Crawler and Indexer

The crawler component employs a breadth-first search (BFS) algorithm to systematically discover and index web pages starting from an initial seed URL.

Key functionalities include:

* HTML Parsing: Utilizes Jsoup library for robust extraction of textual data, effectively handling malformed HTML structures.
* Tokenization Pipeline: Implements custom tokenization strategies optimized for efficient textual content processing.
* Stop Word Removal and Porter Stemming: Enhances indexing accuracy and search recall by standardizing and reducing lexical variations.
* Inverted Indexing: Maintains separate inverted indexes for title and body texts, enabling fast retrieval based on query relevance.
* Incremental Crawling: Implements intelligent mechanisms to update the index incrementally by reprocessing only modified pages, significantly improving crawling efficiency.
* N-gram Indexing: Supports precise phrase searches with indexed n-grams of up to three consecutive words.

Robust error handling strategies manage network issues, cycle detection, and character encoding problems, ensuring uninterrupted operations and continuous availability.

1.3 Search Engine Component

The search engine employs advanced algorithms designed to accurately rank and retrieve relevant documents from extensive datasets.

Features include:

* Vector Space Model: Utilizes TF-IDF weighting, normalized against maximum term frequency, providing balanced scoring across documents.
* Cosine Similarity: Computes document-query similarity effectively, normalizing document lengths for consistent relevance ranking.
* Title Field Boosting: Significantly improves search accuracy by assigning higher weighting (fivefold) to terms found in document titles compared to those in body content.
* Indexed Phrase Searches: Enables precise multi-word query handling, enhancing search specificity and user satisfaction.
* Performance Optimizations: Implements caching mechanisms, precomputed indices, and early termination conditions, drastically reducing query response times.
* Command-Line Interface: Provides an intuitive interactive interface complemented by robust error handling, facilitating user-friendly query operations.

1.4 Integration and Communication

Communication between crawler and search engine components is managed through well-defined interfaces, promoting seamless integration, flexibility for updates, and component independence. Persistent storage and rapid retrieval operations are achieved through JDBM, ensuring data consistency and operational robustness.

2. Database File Structures

2.1 Forward Indexes

* pageIdToUrl: Efficiently maps page IDs to URLs for quick retrieval and accurate presentation of search results.
* pageInfo: Stores comprehensive metadata, including titles, modification dates, file sizes, and linked pages, enhancing data retrieval precision.
* wordIdToWord: Supports fast reverse mapping from numeric identifiers to corresponding stemmed words.

2.2 Inverted Indexes

* bodyInvertedIndex: Stores mappings of terms to document frequencies specifically within document bodies.
* titleInvertedIndex: Maintains a separate mapping of terms to document frequencies within titles, providing targeted indexing capabilities.

2.3 Mapping Tables

* urlToPageId and wordToWordId: Facilitates efficient bi-directional lookup and ensures indexing consistency across the system.

2.4 Phrase Search Structures

* pageIdToBodyWords and pageIdToTitleWords: Maintains sequential word information, significantly optimizing exact phrase search queries.

2.5 Optimization Structures

* maxTFForPageId: Precomputes and stores maximum term frequencies for each document, accelerating normalization during relevance scoring.
* counter: Ensures unique identifiers for both pages and terms, maintaining data integrity and consistency across indexing and retrieval operations.

3. Algorithms

3.1 Web Crawling Algorithm (BFS)

The BFS algorithm systematically explores web pages in a layered manner, beginning from a seed URL. It prioritizes proximity and minimizes redundancy, employing cycle detection, and politeness policies to maintain optimal server load and effective indexing.

3.2 Text Processing Pipeline

* HTML Extraction (Jsoup): Ensures accurate and structured data extraction.
* Tokenization and Normalization: Efficiently converts extracted text into index-ready tokens.
* Stop-word Removal and Porter Stemming: Refines indexing and enhances retrieval accuracy.
* Indexed N-grams: Optimizes response times and enhances search query accuracy.

3.3 Vector Space Model

Documents and user queries are modeled as vectors in a high-dimensional space, weighted using TF-IDF. Cosine similarity accurately computes the relevance, providing uniform scoring independent of document length.

3.4 Title Field Boosting

Assigning greater weight to title terms effectively prioritizes more relevant documents, significantly enhancing user experience and satisfaction with the search results.

3.5 Phrase Search

Employs a dual-phase strategy, initially identifying candidate documents based on individual term matches, followed by rigorous sequence verification to ensure precise retrieval of exact phrases.

3.6 Performance Optimizations

Performance enhancements include sophisticated caching strategies, precomputations, and strategic early termination, collectively ensuring sub-second query responsiveness.

4. Advanced Features

* N-gram Indexing: Enhances accuracy for phrase-based searches.
* Incremental Crawling: Dramatically reduces redundant data processing.
* Title Boosting: Markedly improves relevance and accuracy of search results.
* Query Optimizations: Ensures rapid and efficient query handling.
* Robust Error Handling: Provides continuous stability and operational resilience.

5. Testing Results

5.1 Crawler Testing

* Demonstrated comprehensive efficiency and robustness across diverse web environments.
* Verified cycle detection mechanisms and effective incremental updates.
* Successfully indexed over 300 pages, achieving high accuracy in varied conditions.

5.2 Search Engine Testing

* Confirmed precise relevance and accuracy across diverse query scenarios including single-term, multi-term, and phrase queries.
* Validated performance optimizations through rapid response times, significantly enhancing the user experience.

6. Conclusion

6.1 Strengths

* Efficient storage and retrieval operations via JDBM.
* Improved result accuracy and relevance through sophisticated title boosting.
* Effective optimization strategies ensuring rapid, reliable user interactions.

6.2 Weaknesses

* Limited coverage due to BFS crawling.
* Phrase indexing constrained to trigrams.
* Lack of immediate result context due to absent snippet generation.
* Basic TF-IDF implementation limited compared to advanced ranking models.
* Scalability challenges with very large datasets.

6.3 Recommendations

* Adoption of scalable libraries such as Lucene.
* Implementation of advanced ranking algorithms like BM25.
* Extension of phrase indexing capabilities beyond trigrams.
* Incorporation of snippet generation for improved user assessment.
* Enhanced crawling strategies for more comprehensive web coverage.

6.4 Potential Innovations

* Integration of semantic query expansion technologies.
* Implementation of authority-based ranking models (e.g., PageRank).
* Development of personalized search result capabilities.
* Expansion into multimedia indexing.
* Incorporation of cross-language search functionalities.

7. Team Contributions

Equitable contributions included:

* Comprehensive implementation and rigorous testing of crawling and indexing modules.
* Sophisticated development of search algorithms and interactive interfaces.
* Extensive database design, detailed documentation, and thorough system optimization.

Collaborative efforts significantly contributed to the successful execution and robust performance of the search engine system, highlighting both teamwork effectiveness and technical proficiency.