Salary Stratos
-top paid job analysis

110127749 - Deon Victor Lobo (Team Lead)
110123330 - Gagandeep Singh
110126400 - Rishit Devang Bhojak
110128309 - Rahul Harish Patel
110124682 - Smit Hiteshkumar Patel

Introduction

- **High-Paying Job Analysis:** Uncover insights from top job listing sites.
- **Clear Search Rules:** Efficient data retrieval for meaningful patterns.
- **User-Friendly Interface:** Easy keyword-based job search for diverse users.
- **Cross-Site Results:** Aggregates data from Simply Hired, Remote Ok, Glass Door.
- **Page Ranking:** Highlights relevant and high-paying jobs systematically.
- **Valuable Industry Insights:** Benefits job seekers, employers, and industry enthusiasts.

Web crawler

- **Web Crawling:** Utilize Selenium for dynamic web page interaction.
- **Data Structures:** Queue for job links, HashSet for unique identifiers.
- **Algorithms:** Jsoup for parsing HTML and extracting job links.
- **Efficiency:** Prioritize unique jobs, optimize link processing order.
- **Scraping Approach:** Selenium for SimplyHired and GlassDoor, direct extraction for RemoteOK.
- **JSON Storage:** Save validated job data in JSON format.

Rishit Devang Bhojak

Finding patterns using regular expressions

- **Regex Power:** Leverage regular expressions for salary pattern extraction.
- **Versatile Parsing:** Handle diverse formats with conditional logic and Regex.
- **Numeric Extraction:** Utilize replaceAll for numeric value extraction.
- **Consistent Conversion:** Convert salaries to yearly format with precision.
- **Pattern Matching:** Identify salary structures using crafted Regex patterns.
- **Enhanced Accuracy:** Ensure uniform representation for meaningful analysis.

Rishit Devang Bhojak

HTML Parser

- **Jsoup Mastery:** Leverage Jsoup library for HTML parsing.
- **Data Extraction:** Retrieve title, company, location, and salary.
- **Regex Precision:** Utilize regex for varied salary formats.
- **Structured Output:** Create Job objects for organized data.
- **Web Scraping Precision:** Extract details for further processing.
- **Jsoup Advantages:** Convenient HTML parsing for efficient data retrieval.

Smit Hiteshkumar Patel

Frequency count

- **Trie Powerhouse:** Leverage Trie for efficient word-based data retrieval.
- **TrieNode Insight:** TreeMap stores word frequencies for detailed analysis.
- **Frequency Counts:** TreeMap in TrieNode facilitates word frequency tracking.
- **Inverted Indexing:** Trie structure enables quick job retrieval by words.
- **Sorting Capability:** Facilitates sorting words based on their frequencies.
- **Efficient Data Analysis:** Trie enhances search speed in large datasets.

Smit Hiteshkumar Patel

Inverted Indexing

- **Efficient Inverted Indexing:** Trie structure for quick word-based searches.
- **Stop Word Filtering:** HashSet removes common English stop words.
- **TrieNode Intelligence:** Nodes store job IDs and word occurrences.
- **Streamlined Prefix Searches:** Trie facilitates efficient word retrieval.
- **Dynamic Index Construction:** Trie adapts to diverse job descriptions.
- **Optimized Data Structure:** Trie enhances speed in large datasets.

Page Ranking

- **Efficient Search Retrieval:** TrieDS used for quick search term data retrieval.
- **Dynamic Sorting:** SortedArray maintains jobs sorted by word frequencies.
- **Binary Search Optimization:** Fast insertion based on frequency using binary search.
- **Cumulative Word Frequencies:** Ranks jobs based on search term occurrences.
- **Page Ranking Algorithm:** Iterates through search terms for effective ranking.
- **Dynamic Ranking Updates:** Ensures accurate and dynamic job ranking.

Common Functions

- **Efficient Link Extraction:** Scrapes job links for individual crawling.
- **Singleton for Chrome Driver:** Single ScraperBot instance ensures consistent Chrome Driver.
- **Trie for Text Indexing:** Created Trie Data Structure
- **Stopword Filtering:** Trie ignores common English stopwords during insertion.
- **SortedArray for Ranking:** Utilizes sorted array with comparator for efficient sorting.
- **Algorithmic Efficiency:** Employs binary search for fast data structure operations.

Spell Checking

- **Dictionary:** SpellChecker utilizes a HashMap for efficient word storage.
- **Edit Distance Algorithm:** Calculates word similarity for spell-checking accuracy.
- **Frequency Map:** TreeMap stores word frequencies for suggestion prioritization.
- **Dictionary Initialization:** Loads valid words from a file and improves with scraped data.
- **Nested TreeMap Structure:** Organizes suggestions by edit distance and word frequency.
- **Effective Spell-Checking:** Robust functionality with Trie, Edit Distance, and frequency data structures.

Gagandeep Singh

Word Completion

- **Trie-Powered Suggestions:** WordCompletion uses TrieDS for efficient word retrieval.
- **Prefix Search Algorithm:** Trie's searchInTrieWithPrefix fetches words based on prefixes.
- **Context-Aware Sorting:** Suggestions are organized using a SortedArray by frequency.
- **Responsive Suggestions:** Trie-based approach provides quick and relevant word suggestions.
- **Enhanced Search Terms:** Validate and generate suggestions for validated search terms.
- **Effective Word Completion:** Trie and SortedArray ensure efficient and context-aware suggestions.

Gagandeep Singh

Recommend top paid jobs based on search results

- **Salary-Based Sorting: Trie and SortedArray for efficient job ranking.**
- **Inverted Indexing:** TrieDS structures job terms for quick retrieval.
- **Cost Calculation:** Combines salary and frequency for optimal ranking.
- **Descending Order:** SortedArray maintains descending order of job costs.
- **Efficient Discovery:** Rapid identification of relevant top-ranking jobs.

Gagandeep Singh

Common Functions

- **Jsoup Element Extraction:** RemoteOk.java uses Jsoup for scraping web elements.
- **JSON Database:** ScrapperBot saves and reads job data from database.json.
- **Model for Job Data:** Jobs.java defines a model for storing job details.
- **Spring-React Integration:** FeatureController integrates Spring Boot APIs with React UI.

Data validation using regular expressions

- **AtomicBoolean for Thread Safety:** Ensures synchronized updating of validation status.
- **Regex for Field Validation:** Employs regular expressions for pattern-based data validation.
- **Comprehensive Data Check:** validateDataForOneObject ensures all job fields undergo validation.
- **Data Quality Assurance:** Filters and adds only validated jobs to the database.
- **Thread-Safe Validation:** AtomicBoolean guarantees safe multi-threaded validation updates.
- **Efficient Field Validation:** Regex patterns enhance accuracy in validating job information.

Deon Victor Lobo

Search Frequency

- **LRUCache for Efficient Storage:** Custom LRUCache maintains recent search term frequencies.
- **SortedArray for Display:** Utilizes a sorted array for presenting search terms.
- **Dynamic User Insights:** LRUCache and sorting offer real-time search patterns.
- **Limited Cache Size:** Ensures efficient memory use with a 50-word limit.
- **User Interaction Tracking:** Frequencies updated based on user search terms.
- **Responsive UI Data:** Provides top 10 search terms and frequencies.

Deon Victor Lobo

Compare sorting algorithms for Page ranking

- **PageRanking Integration:** Compare different sorting algo for page ranking
- **Merge Sort**
- **Binary Search**
- **QuickSort**
- **Efficiency Assessment:** Evaluate runtime differences for sorting algorithms.
- **Binary Search Advantage:** Faster in partially sorted data scenarios.

Deon Victor Lobo

Thank you