SEARCHX: An Integrated Framework of Distributed Intelligent Search Services Based on Web Browser

Zehui Zhang*, Lin Zhou, Jie Peng, Liwei Wang, Bo Cheng
Inner Mongolia Power Digital Research Institute, Hohhot 010000, China

***Abstract*—To address the growing demand for web search and improve the performance and accuracy of search systems, this study proposes a distributed intelligent search service integration framework based on SEARCHX. This framework leverages the local computational power of the browser, integrating inverted indexing, data sharding, and replication mechanisms, as well as the Term Frequency-Inverse Document Frequency (TF-IDF) intelligent ranking algorithm. These components enable front-end distributed processing of search tasks and multi-source result fusion. Experiments are conducted on six major browser platforms, Chrome, Firefox, Edge, Safari, etc., using the open-source Text REtrieval Conference (TREC) dataset. The system's response performance and accuracy are evaluated under varying search loads. The experimental results show that, compared to the unoptimized version, the optimized SEARCHX reduces the average response time by approximately 27 per cent under medium-to-high load conditions. Precision improves by an average of 0.05, and the F1 score increased by more than 0.04 on all platforms. The system also demonstrates good stability and consistency across multiple platforms. SEARCHX provides a viable approach to building decentralized, high-efficiency, and easily deployable intelligent search services, with strong practical value and expansion potential. This study aims to construct a decentralized, cross-platform, and high-performance intelligent search service framework, offering a more efficient, stable, and accurate technical support solution for users in complex search environments.**

Keywords*—Web; TF-IDF; distributed network; intelligent search; SEARCHX

I. Introduction

In the context of rapid advancements in information technology, internet data is experiencing exponential growth, and users face unprecedented challenges of information overload and retrieval difficulties in the web environment. As the core tool for information access, intelligent search engines have permeated multiple fields such as knowledge acquisition, data mining, and personalized recommendation systems [1]. However, mainstream search engines are predominantly based on centralized server architectures. While they offer powerful computational capabilities and global indexing services, they increasingly show issues in handling massive concurrent requests, ensuring user privacy, and supporting flexible deployments. These problems include high latency, poor scalability, high deployment costs, and data security risks [2]. Meanwhile, web browser technology is undergoing significant evolution. New-generation browser-side technologies, represented by JavaScript engine optimizations, WebAssembly (Wasm), and Service Workers, have transformed web browsers from mere information display tools into powerful computing platforms [3]. The enhanced local storage, local execution, and multi-threading capabilities of browsers provide a practical foundation for running high-performance applications directly within the browser. This shift enables parts or even the entire search process to be offloaded to the browser, which alleviates server load, reduces response latency, enhances data localization capabilities, and strengthens user privacy protection [4]. In enterprise-level information systems, distributed search architectures like Elasticsearch are widely deployed, relying on mechanisms such as inverted indexing, data sharding, and replica redundancy to improve search efficiency and system fault tolerance [5]. However, these technologies are still primarily server-side and have not been effectively migrated or integrated into the browser side, leaving a gap in research and practical application [6]. Therefore, fully leveraging the potential of browsers as client-side computing platforms to construct a cross-platform, intelligent, distributed search framework becomes an important and worthy direction for further exploration.

Based on the above background, this study proposes the SEARCHX framework, a distributed search engine optimization solution that integrates inverted indexing, data sharding, replica synchronization, and the Term Frequency–Inverse Document Frequency (TF-IDF) intelligent ranking algorithm. The framework is introduced within the browser environment to enable distributed task scheduling, plugin-based service integration, and local search result fusion. This design supports cross-platform deployment and flexible scalability, and significantly enhances system response efficiency and user data privacy control capabilities. The main innovation of this study is to systematically migrate the complete distributed search engine architecture—including inverted index, data sharding, replica synchronization, and TF-IDF intelligent ranking algorithm—to the Web browser side. This design is expected to realize truly decentralized, cross-platform search processing and local multi-source result fusion, thus filling the gap in the research and practice of a complete distributed search architecture on the browser side.

The subsequent structure of this study is arranged as follows: Section II reviews the related work on intelligent search, distributed architecture, and browser computing. Section III elaborates on the design of the SEARCHX framework and distributed optimization methods in detail. Section IV presents the performance evaluation and comparative experimental results under multi-browser platforms. Section V summarizes the research results and looks forward to future directions. Through systematic discussion and empirical analysis, this study aims to comprehensively present the technical implementation

*Corresponding author.

and performance advantages of the SEARCHX framework to readers.

II. Related Work

A. Distributed Intelligent Search Algorithm

In recent years, both academia and industry have conducted extensive research on intelligent search systems, distributed architectures, and front-end computing, resulting in several valuable development paths [7]. In the field of intelligent search, Wang et al. discovered that pre-trained encoders such as Bidirectional Encoder Representations from Transformers (BERT) were mainstream methods for information retrieval. BERT could be categorized into six technical paths, including long document processing, semantic integration, and efficiency balancing. Their results indicated that BERT-based models outperformed traditional methods in terms of accuracy and adaptability, with lower computational costs [8]. He et al. proposed that personalized recommendation mechanisms could enhance search user satisfaction and result relevance by modeling user historical behaviors and interest preferences [9]. In the domain of distributed search systems, Fan et al. found that inverted index structures significantly improved recall and precision in large-scale text retrieval, making it a core technology for building high-performance search engines [10]. Merlin and Prem identified that the MapReduce framework based on the Jaya-Sine Cosine Algorithm (Jaya-SCA) enabled efficient indexing and retrieval in big data. This method achieved an F1 score of 0.5323, a recall rate of 0.4400, and a precision rate of 0.6867 on the StatLog heart disease dataset, significantly improving information retrieval performance [11]. Soltanmohammadi et al. found that a consistent hashing-based data partitioning strategy effectively avoided data loss due to node failures [12].

B. Browser-Side Computing and Search Results Sorting

Regarding browser-side computing and collaboration mechanisms, Kjorveziroski and Filiposka highlighted that, with the development of WebAssembly, modern browsers had the ability to execute local computational tasks, showing potential to serve as edge nodes in distributed systems [13]. Putra et al. designed the SearchX platform, which supported user behavior tracking and collaborative task allocation, demonstrating that browsers could serve as both execution and collection terminals for search services. However, their system mainly focused on research purposes and lacked complete distributed indexing and search capabilities [14]. In ranking algorithms, Dai et al. found that while the TF-IDF algorithm was simple, it effectively reflected keyword importance and document relevance in static document collections, serving as the foundation for search ranking [15]. Yang and Choi discovered that the Best Matching (BM) 25 algorithm, by introducing a document length normalization factor, further improved ranking accuracy [16]. In search service integration systems, Dejonckheere found that search service platforms based on microservice architectures effectively enhanced module decoupling and deployment flexibility, widely applied in cloud computing and large-scale web services [17].

In summary, although significant progress has been made in areas such as distributed search, intelligent ranking, and browser-side computing, there is currently a lack of a unified integrated platform that can simultaneously support local browser computation, cross-node collaborative search, and intelligent result fusion. The SEARCHX framework proposed here builds upon the aforementioned research achievements and fills the gap in browser-side distributed search integration. This framework holds significant theoretical and practical value.

III. Research Methodology

A. SEARCHX Framework

SEARCHX is a distributed intelligent search service integration framework based on modern web browsers, designed to leverage the computational and communication capabilities of browsers to provide a low-barrier, high-performance collaborative search experience. Its architecture adopts a client-server model, with the front-end implemented using the React framework for modular and component-based design. It supports multi-browser cross-platform access without the need for additional software installations. Users can simply participate in search tasks by specifying a Uniform Resource Locator (URL). The front-end is responsible for displaying the user interface, managing search sessions, and collecting user operation logs in real-time. The back-end, based on the Node.js platform, uses Express and Socket.io to handle efficient Hypertext Transfer Protocol (HTTP) requests and WebSocket real-time communication, managing user grouping, task synchronization, and data storage [18]. For database management, SEARCHX utilizes MongoDB to support dynamic data structures, enabling flexible storage of massive logs and experimental data. The basic framework of SEARCHX is illustrated in Fig. 1.

B. SEARCHX Framework Based on Distributed Search Optimization

To meet the high-performance requirements of massive data searches and enhance the accuracy and stability of the system, the SEARCHX framework introduces several distributed search optimization techniques on the browser side. These techniques include the construction of inverted indexes, the TF-IDF ranking algorithm, data sharding and replication mechanisms, as well as task scheduling and result fusion based on real-time communication.

Inverted indexing is the core data structure of modern search engines. It establishes a mapping from keywords to document lists, enabling fast keyword location and retrieval. In a distributed environment, local construction of inverted indexes helps improve search speed and the system's parallel processing capabilities [19]. Specifically, the inverted index is composed of a keyword and the inverted list of documents. The calculation equation is as follows:

$$I = \{ (t_i, L_i) \mid t_i \in T \} \tag{1}$$

$$L_i = \{ (d_j, f_{ij}) \mid d_j \in D, f_{ij} > 0 \} \tag{2}$$

I is an inverted index set. t_i is the i-th keyword in the keyword set T. L_i is the inverted list corresponding to the keyword t_i. d_j is the j th document in the document set D. f_{ij} is the word frequency of keyword t_i in document d_j.

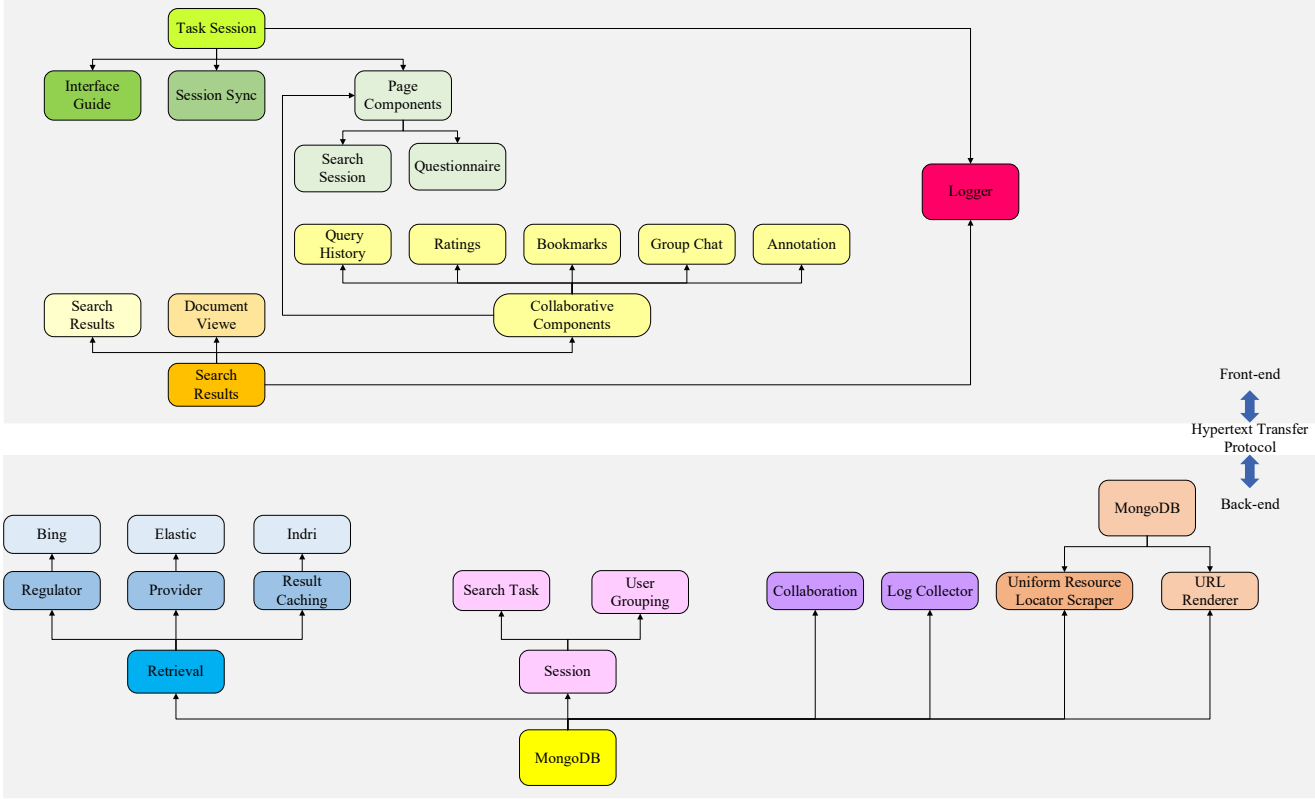

Fig. 1. SEARCHX basic framework.

The TF-IDF algorithm is a classical weighting method based on the vector space model, which combines the frequency of keywords in a single document and their rarity in the whole document set, thus weighting keywords and highlighting important and highly differentiated words [20-22]. The specific calculation equation is as follows:

$$tf_{ij} = \frac{f_{ij}}{\max_{t_k \in d_j} f_{kj}} \tag{3}$$

$$idf_i = \log\left(\frac{N}{df_i + 1}\right) \tag{4}$$

$$TF - IDF(t_i, d_j) = tf_{ij} \cdot idf_i \tag{5}$$

tf_{ij} is the normalized word frequency. idf_i is the inverse document frequency. df_i is the number of documents containing the keyword t_i. N is the total number of documents.

The vector space model represents documents and queries as multidimensional vectors, and quantifies the correlation between them by calculating the cosine similarity of the included angle between the vectors [23]. This measurement method not only overcomes the rigid limitation of the traditional Boolean model but also flexibly reflects the similarity between texts and improves the accuracy of search matching. The calculation equation of the vector space model is as follows:

$$\mathbf{v}_{d_j} = (v_{j1}, \ldots, v_{jM}) \tag{6}$$

$$v_{ji} = TF - IDF(t_i, d_j) \tag{7}$$

$$\mathbf{v}_Q = (w_1, \ldots, w_M) \tag{8}$$

$$w_i = \begin{cases} w_i, & t_i \in Q \\ 0, & otherwise \end{cases} \tag{9}$$

$$Sim(d_j, Q) = \frac{\mathbf{v}_{d_j} \cdot \mathbf{v}_Q}{\|\mathbf{v}_{d_j}\| \cdot \|\mathbf{v}_Q\|} \tag{10}$$

$\mathbf{v}_{d_j}$ is the vector representation of document d_j. $\mathbf{v}_Q$ is the vector representation of query Q. M is the total number of keywords. w_i is the weight of keyword t_i in the query. $Sim(d_j, Q)$ is the similarity value between the document and query.

Sharding is a commonly used data partitioning method in distributed systems, where large-scale data is horizontally split across multiple nodes to reduce the load on individual nodes and enable parallel processing. A hash function ensures that data is evenly distributed, preventing load imbalance across nodes [24]. The sharding mechanism is a key technology for achieving system scalability and handling high concurrency. The specific partitioning expression is as follows:

$$S_m = \{d_j \in D \mid h(d_j) mod K = m\} \tag{11}$$

S_m is the m-th slice. $h(d_j)$ is the hash value of document d_j. K is the total number of slices. m is the slice number.

The replication mechanism enhances the fault tolerance and availability of the system by storing identical data across multiple nodes. In the event of a node failure, the system can switch to a backup replica, ensuring continuous and stable

operation [25]. Additionally, replication helps with load balancing and improves access efficiency. The replica selection algorithm can be expressed as follows:

$$R_m = \{n_{m1}, n_{m2}, \ldots, n_{mR}\} \tag{12}$$

$$n^* = arg\min_{n \in R_m} Load(n) \tag{13}$$

R_m is the copy set of the m-th slice. n_{m1} is the replica node. $Load(n)$ is the current load value of node n. n^* is the selected replica node.

Parallel computing divides large tasks into smaller sub-tasks and assigns them to different computation nodes for simultaneous execution, reducing the overall response time. Search tasks are split into shard queries, and high-efficiency collaboration between nodes is achieved through modern Web asynchronous communication protocols [26]. The system response time can be calculated as follows:

$$T_{response} = \max_{0 \le m < K} T_m \tag{14}$$

$T_{response}$ is the overall response time of the system. T_m is the query response time of the m-th fragment.

Multi-source result fusion, based on data fusion theory, integrates partial results from each shard, eliminates duplicates, and forms the final ordered result set [27]. Merge sorting ensures the correctness of the overall ranking, while deduplication prevents result redundancy, enhancing the user experience. The specific calculation equation is as follows:

$$R = \bigcup_{m=0}^{K-1} R_m \tag{15}$$

$$R_{final} = \delta\varepsilon\upsilon\pi\lambda\iota\chi\alpha\tau\varepsilon(R) \tag{16}$$

R is the combination result set. R_{final} is the result set after final sorting and deduplication.

The load balancing theory provides a dynamic adjustment strategy for task allocation. By monitoring the node load in real-time and combining it with the weighted smoothing algorithm, the task scheduling weight is adjusted to prevent the overload of a single node and improve the system stability and resource utilization [28]. The dynamic updating equation of scheduling weight is as follows:

$$w_n^{(t+1)} = \alpha w_n^{(t)} + (1-\alpha)\frac{1}{Load(n)} \tag{17}$$

The maximum concurrency control calculation equation is as follows:

$$C = min\left(\frac{X}{Tt}, C_{max}\right) \tag{18}$$

$w_n^{(t)}$ is the scheduling weight of node n at time t. α is a smoothing factor (0~1). X is the total number of tasks to be processed. Tt is the time window size. C_{max} is the maximum number of concurrent tasks supported by the system.

The process of SEARCHX based on distributed search optimization is shown in Fig. 2.

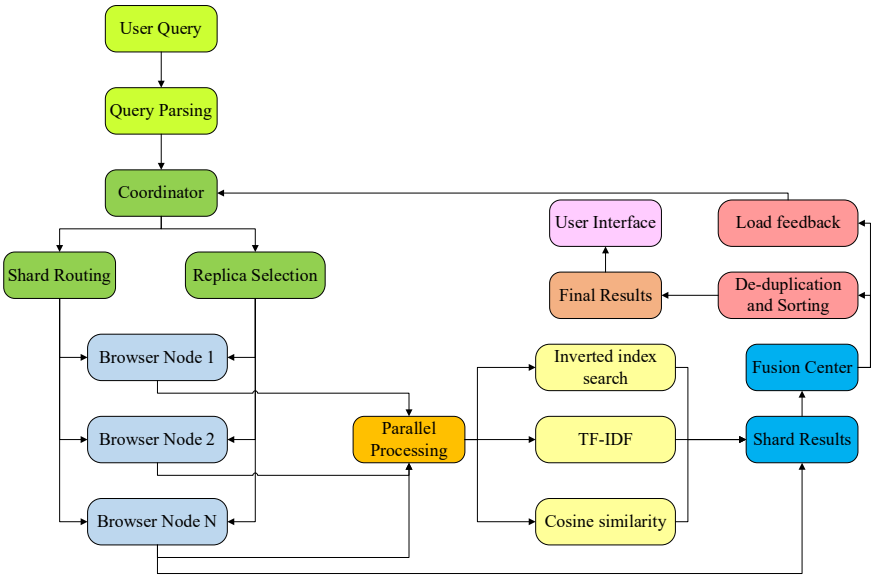

Fig. 2. The process of SEARCHX based on distributed search optimization.

C. System Assembly Design

The experimental platform of this study consists of six identical physical computers, each equipped with an Intel Core i7-10700 CPU, 6GB of RAM, and 512GB solid-state drive storage. The operating system on all machines is Linux Ubuntu 20.04 Long-term Support (LTS). The latest versions of mainstream web browsers, including Chrome, Firefox, Edge, Safari, Opera, and Brave, are deployed on each node to test the compatibility and performance differences of SEARCHX in a multi-platform browser environment. The software development environment includes Node.js v18.0 for the backend service runtime, React v18.2 for frontend component development, and MongoDB v6.0 for lightweight local indexing and data storage. Additionally, the system communication layer incorporates WebSocket and Web Real-Time Communications (WebRTC) protocols to support distributed collaborative search across browsers.

SEARCHX adopts a modular and component-based system deployment strategy. The main node runs the backend service responsible for task scheduling and user sessions, while the browser nodes act as independent search sub-nodes that load the frontend interface and locally execute indexing and query tasks. The deployment process includes the following steps: 1) Configure Node.js and Express backend services on the main node and open the WebSocket communication port. 2) Run the frontend page on each browser node, loading the SEARCHX interface via a unified URL. 3) Each browser node autonomously registers and identifies its role based on the system initialization logic. 4) Use browser-side Service Workers and localStorage to achieve local data persistence and offline accessibility. This architecture allows the browser to function both as a lightweight client for displaying the interface and as a search computation node, enabling a truly decentralized search system at the edge.

The testing plan is divided into two stages: functional testing and performance testing. In functional testing, the system verifies the functionality of critical modules, including: task scheduling, index construction, result sorting, node communication, and failover mechanisms. The goal is to ensure that these modules are operational and functioning as expected within the system. In performance testing, the system is tested under different search request volumes (10, 100, 500, 1000, 2000, 5000 requests). The tests are conducted on six major browser platforms, and the system's response time, stability, and other performance metrics are evaluated. Each node automatically logs data, which is sent back to the main node for analysis and assessment.

This study uses the Text REtrieval Conference (TREC) open text retrieval dataset as the training and evaluation data source for the system [29, 30]. This study selects this dataset mainly based on the following considerations: the dataset includes 250 real user queries, more than 520,000 news documents, and manually annotated relevance judgments. It features diverse query topics, moderate document scale, and high annotation quality, which can comprehensively evaluate the precision, recall, and system stability of the search engine under different query complexities and document scales, and meet the needs of this study for empirical verification of retrieval performance. The subset used in this study is the TREC 2004 Robust Track, which includes 250 standard query tasks, 528,000 English news and article documents, and manually annotated relevance grade labels. This makes it suitable for analyzing precision, recall, accuracy, and F1 score metrics.

IV. Experimental Evaluation of SEARCHX Performance

A. Results of Performance Analysis

The comparison results of the performance changes of the search framework under different platforms with the query volume are shown in Fig. 3 to Fig. 8.

Analysis of Fig. 3 shows that, as the number of query entries increases, the average response time for all browsers tends to rise, indicating that an increase in search request volume has a significant impact on system response. Among all browsers, Chrome performs the best, with the shortest response time, suggesting that its JavaScript engine and network communication optimizations are more suitable for distributed search loads. Safari, on the other hand, performs relatively poorly, with the longest response time. The increase in response time accelerates as the query volume grows, and for 5000 queries, the average response time is approximately 6 to 7 times higher than that for 10 queries. It indicates that the system can still maintain relatively stable response performance even under large-scale tasks.

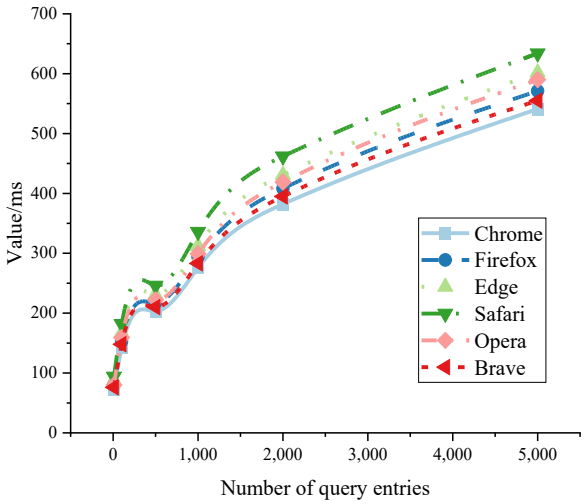

Fig. 3. Comparison results of average response time.

Fig. 4 shows that the CPU utilization increases significantly with the growth of query volume, reflecting the increasing computational resource consumption of search tasks. Chrome and Brave demonstrate lower CPU usage under varying query loads compared to other browsers, indicating more efficient resource management. Safari and Edge exhibit the highest CPU utilization under heavy loads, which could potentially affect their ability to handle multiple tasks concurrently. Overall, in distributed tasks, it is essential to allocate the load reasonably in this system to avoid CPU bottlenecks on a single node.

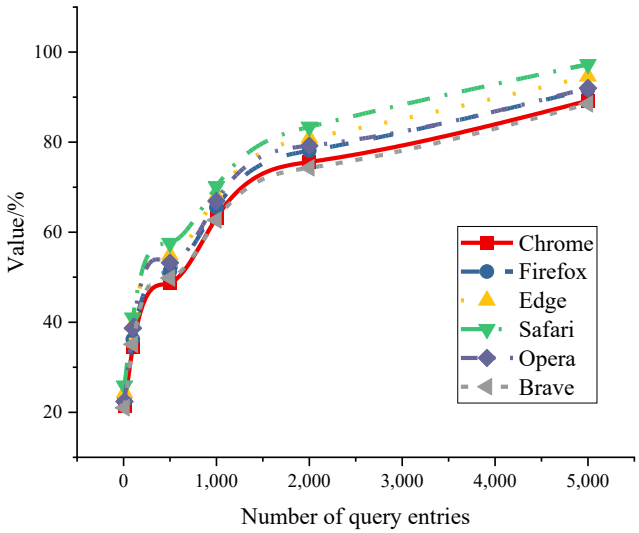

Fig. 4. CPU occupancy comparison results.

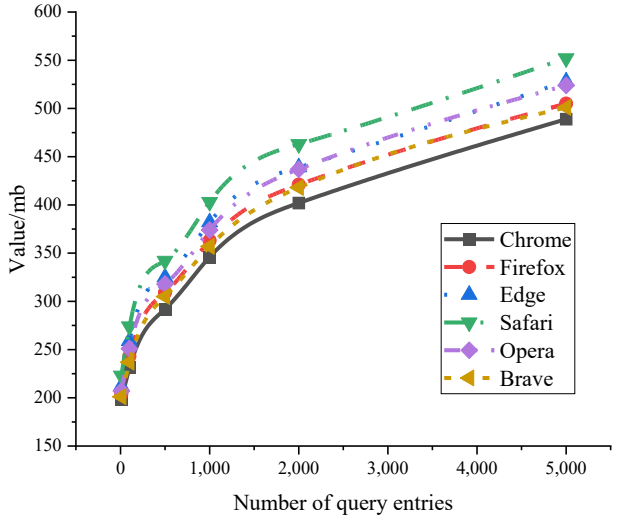

Fig. 5. Memory usage comparison result.

Fig. 5 reveals that memory consumption in all browsers increases linearly with the number of query entries, indicating that search result indexing and cached data consume a significant amount of memory. Chrome and Brave exhibit relatively lower memory usage, suggesting optimized memory management and recycling mechanisms. Safari, on the other hand, shows the highest memory consumption, indicating potential issues with memory leakage or room for improvement in cache management strategies.

Fig. 6 shows that the throughput rate decreases as the query volume increases on all platforms, indicating a certain processing capacity bottleneck. Chrome maintains the highest throughput, handling more requests consistently, which reflects its strong support for distributed search tasks. Edge and Safari, however, exhibit lower throughput, likely due to limitations in network communication efficiency and multi-threading processing capabilities. This result underscores the critical importance of improving network communication and concurrent processing technologies for system performance optimization

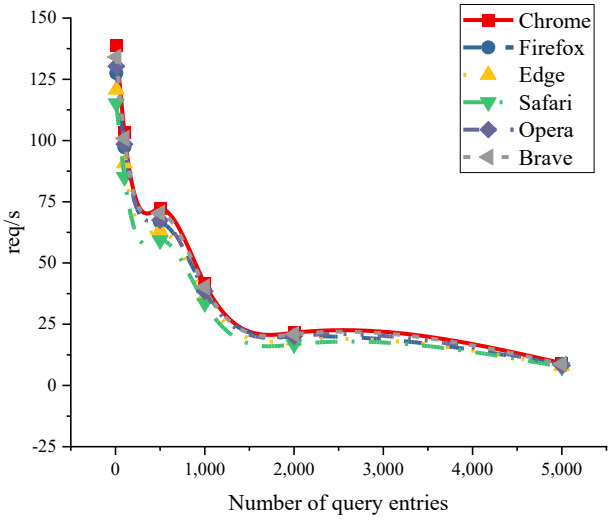

Fig. 6. Throughput comparison results.

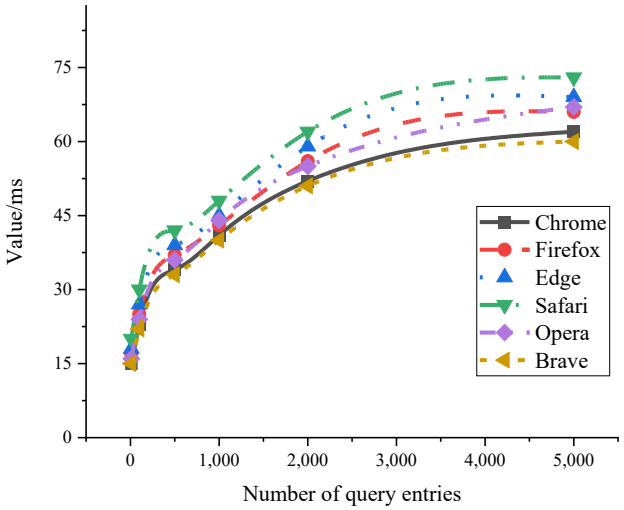

Fig. 7. Data synchronization delay result.

Fig. 7 shows that data synchronization latency increases as the query volume grows, indicating a higher communication burden between nodes. Chrome and Brave maintain the lowest synchronization delay, reflecting superior WebSocket or WebRTC communication performance. Safari, on the other hand, exhibits the highest latency, which could degrade the user experience, particularly in collaborative search scenarios. The study suggests that optimizing the synchronization mechanism is a key factor in enhancing system response speed and stability.

Fig. 8 reveals that the overall score decreases as the query load increases, reflecting the system's stability pressure under high load conditions. Chrome and Brave maintain higher scores than other platforms, indicating better stability in complex load environments. Edge and Safari have lower scores. It suggests that future optimization of the SEARCHX framework should focus on improving error recovery and resource management strategies to ensure stable operation during prolonged high-load conditions.

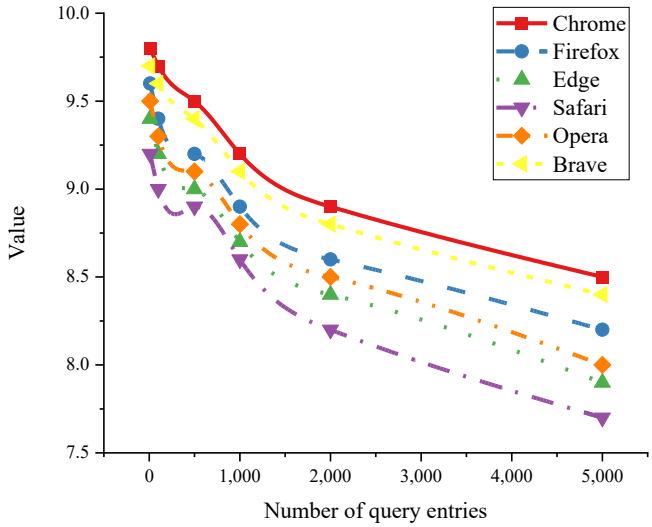

Fig. 8. System stability score results.

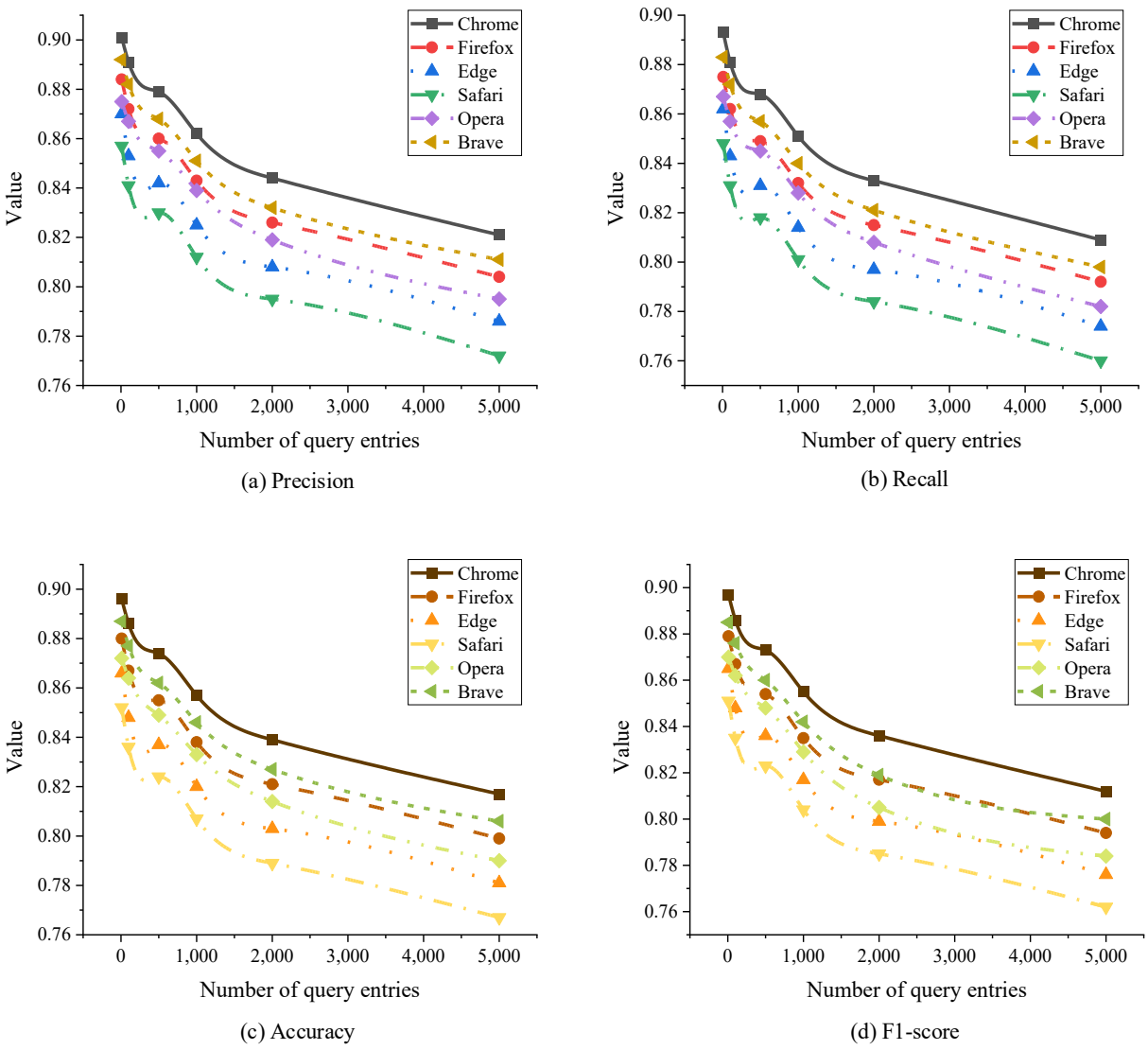

Fig. 9. Comparison of query accuracy under different platforms.

The comparison of query accuracy under different platforms as the query volume increases is shown in Fig. 9.

Fig. 9 shows that as the number of query entries increases from 10 to 5000, the search result accuracy across all browser platforms exhibits a gradual decline. Taking Chrome as an example, the accuracy drops from 0.901 to 0.821, but it remains at a high level, indicating excellent performance in intelligent ranking and index optimization. Firefox and Brave follow closely, with accuracy dropping from 0.884 and 0.892 to 0.804 and 0.811, respectively. Safari and Edge show slightly weaker performance, starting at 0.857 and 0.870, dropping to 0.772 and 0.786, but still maintaining stability. Regarding recall rate, Chrome and Brave platforms decrease from 0.893 and 0.883 to 0.809 and 0.798, demonstrating strong document coverage and distributed indexing effects. Firefox and Opera decrease from 0.875 and 0.867 to 0.792 and 0.782. Safari and Edge platforms start lower, at 0.848 and 0.862, and drop to 0.760 and 0.774, indicating room for improvement in shard synchronization and index coverage for these two platforms. As the query load increases, the accuracy of all platforms gradually decreases but remains stable. Chrome's accuracy drops from 0.896 to 0.817, and Brave's from 0.887 to 0.806, showing the best performance. Firefox and Opera start at 0.880 and 0.872, and end at 0.799 and 0.790, respectively. Safari and Edge show slightly lower accuracy, from 0.852 and 0.866 to 0.767 and 0.781. The F1 scores for all platforms decline as the number of queries increases, but the minimum remains above 0.760. Chrome and Brave show the best overall performance, with F1 scores dropping from 0.897 and 0.885 to 0.812 and 0.800, respectively. Firefox and Opera's F1 scores drop from 0.879 and 0.870 to 0.794 and 0.784, still maintaining a high level. Safari and Edge's F1 scores decrease from 0.851 and 0.865 to 0.762 and 0.776, slightly lower than the others but still stable overall.

The results of performance comparison between the search framework and the unoptimized framework under different platforms are shown in Table I.

Table I shows that the optimized SEARCHX search framework outperforms the unoptimized version across six major browser platforms, demonstrating higher system

performance and search quality. Specifically, the average response time decreases by approximately 27 per cent, while CPU and memory usage significantly decrease, indicating improved efficiency in resource scheduling and task management. Throughput increased by an average of 34.8 req/s, greatly enhancing the system's ability to handle concurrent requests. In terms of accuracy, precision and F1 scores improve by an average of approximately 0.05. The Safari platform shows the most significant improvement, with a precision increase of 0.064, highlighting the stability and practicality of the optimized framework in large-scale distributed query environments.

TABLE I. THE PERFORMANCE COMPARISON RESULTS OF THE SEARCH FRAMEWORK AND UNOPTIMIZED FRAMEWORK UNDER DIFFERENT PLATFORMS IN THIS STUDY

Platform	Framework Version	Avg. Response Time (ms)	CPU Usage (%)	Memory Usage (mb)	Throughput (req/s)	Precision	F1 Score
Chrome	Unoptimized	328	72.4	946	143.7	0.762	0.769
	The proposed model	238	63.1	802	178.5	0.821	0.812
Firefox	Unoptimized	349	74.8	971	139.1	0.743	0.751
	The proposed model	257	65.6	835	168.3	0.804	0.794
Edge	Unoptimized	367	77.2	990	135.2	0.725	0.731
	The proposed model	274	68.5	857	162.4	0.786	0.776
Safari	Unoptimized	382	79.1	1014	129.5	0.708	0.717
	The proposed model	293	70.2	876	157.2	0.772	0.762
Opera	Unoptimized	341	73.6	938	142.1	0.735	0.742
	The proposed model	249	64.9	819	171.3	0.795	0.784
Brave	Unoptimized	330	71.9	925	145.5	0.754	0.761
	The proposed model	241	62.8	798	177.8	0.811	0.8

B. Discussions

This study achieves significant improvements over previous research in multiple dimensions, including performance, system resource utilization, and search quality, by constructing and optimizing the SEARCHX search framework. Unlike previous models that applied BERT for information retrieval, which excels in semantic understanding but is limited by computational resources, this study focuses on a lightweight search architecture with strong deployability. By employing TF-IDF and distributed inverted indexing, the framework reduces average response time, lowers CPU utilization, and optimizes memory consumption across six major browser platforms, while also improving average throughput. In contrast to previous approaches that used optimization algorithms to enhance recall performance, this study shows an average improvement of approximately 0.043 in F1 score and more than 0.05 in precision, demonstrating superior practicality and platform adaptability. Additionally, building on previous browser collaboration models that are limited to laboratory settings, SEARCHX has successfully implemented a real-world framework supporting local computation and multi-node collaboration. In summary, this study inherits the theoretical advancements in search quality enhancement, system fault tolerance, and architecture design from prior literature. It also makes a substantial breakthrough in performance, efficiency, and adaptability by constructing a unified integrated platform and conducting cross-platform experimental validation. This study fills the gap in existing search systems related to front-end collaboration and resource optimization, providing a new paradigm for the edge deployment and platform independence of future intelligent search systems.

V. CONCLUSION

This study systematically evaluates the performance of the optimized SEARCHX search framework across six major browser platforms, demonstrating significant improvements in several key metrics. In terms of performance, the optimized system reduces the average response time from approximately 350ms to 250ms, representing a 27 per cent decrease. CPU utilization drops by nearly 9 per cent (for instance, Chrome went from 72.4 per cent to 63.1 per cent). Memory consumption decreases by an average of over 140MB, reflecting more efficient system resource management. Regarding throughput, the average increase is about 34.8 req/s, with Chrome showing the most significant improvement, from 143.7 req/s to 178.5 req/s, greatly enhancing concurrent processing capabilities. In terms of search quality, the optimized SEARCHX framework achieves an average improvement of over 0.05 in precision, with Safari showing the most substantial increase, from 0.708 to 0.772, a rise of 0.064. The F1 score also improves, with an average increase of 0.04-0.05, effectively balancing system recall and precision. In summary, the optimized SEARCHX framework outperforms in four key areas: response speed, system resource usage, throughput performance, and search quality. The results demonstrate its strong stability, high efficiency, and high accuracy in multi-platform, large-scale query environments, making it highly promising for broad applications. This study selects this dataset mainly based on the following considerations: the dataset includes 250 real user queries, more than 520,000 news documents, and manually annotated relevance judgments. It features diverse query topics, moderate document scale, and high annotation quality, which can comprehensively evaluate the precision, recall, and system stability of the search engine under different query complexities

and document scales, and meet the needs of this study for empirical verification of retrieval performance.

Although this study verifies the excellent performance of the SEARCHX framework in a laboratory environment, its deployment in real network environments still faces many challenges. For example, heterogeneous network conditions and differentiated browser computing capabilities may affect the collaboration efficiency and result consistency among nodes; although local data processing enhances user privacy protection, it also brings new client-side security considerations; in addition, when facing Internet-scale ultra-large datasets, the current architecture still needs further optimization in terms of index construction and update efficiency. Identifying these limitations is crucial for the future development and practical application of the framework.

Based on this, future research will conduct an in-depth exploration along the following directions: 1) Attempt to introduce a dynamic load balancing algorithm to further improve the collaboration efficiency among heterogeneous browser nodes. 2) Explore the integration of lightweight semantic models with the existing TF-IDF algorithm to enhance the ability to understand complex semantic queries. 3) Extend the framework to the mobile browser environment and evaluate its performance, focusing on optimizing the index efficiency under large-scale data, and simultaneously explore the integration of lightweight neural ranking models to improve the ability to understand complex queries.

REFERENCES

[1] Singh B, Wongmahesak K, Chandra S. Content marketing with search engine optimization: Fostering successful businesses in online marketing 6.0. Practical Strategies and Case Studies for Online Marketing 6.0, 2025, 1(1): 17-36.

[2] Mager A, Norocel O C, Rogers R. Advancing search engine studies: The evolution of Google critique and intervention. Big Data & Society, 2023, 10(2): 20539517231191528.

[3] Wang Z, Bu D, Wang N, et al. An empirical study on bugs in JavaScript engines. Information and Software Technology, 2023, 155(1): 107105.

[4] Makrydakis N. SEO mix 6 Oâs model and categorization of search engine marketing factors for websites ranking on search engine result pages. Int. J. Res. Mark. Manag. Sales, 2024, 6(1): 18-32.

[5] Ladanavar S M, Kamble R, Goudar R H, et al. Enhancing User Query Comprehension and Contextual Relevance with a Semantic Search Engine using BERT and ElasticSearch. EAI Endorsed Transactions on Internet of Things, 2024, 10(1): 1.

[6] Adnyana I G, Dirgayusari A M, Atmaja K J. Data Visualization for Building a Cyber Attack Monitoring Dashboard Based on Honeypot. Sinkron: jurnal dan penelitian teknik informatika, 2024, 8(4): 2510-2518.

[7] Tkachenko O, Chechet A, Chernykh M, et al. Scalable Front-End Architecture: Building for Growth and Sustainability. Informatica, 2025, 49(1): 1.

[8] Wang J, Huang J X, Tu X, et al. Utilizing bert for information retrieval: Survey, applications, resources, and challenges. ACM Computing Surveys, 2024, 56(7): 1-33.

[9] He X, Liu Q, Jung S. The impact of recommendation system on user satisfaction: A moderated mediation approach. Journal of Theoretical and Applied Electronic Commerce Research, 2024, 19(1): 448-466.

[10] Fan J, Khan J, Singh N P, et al. Fulgor: a fast and compact k-mer index for large-scale matching and color queries. Algorithms for Molecular Biology, 2024, 19(1): 3.

[11] Merlin N R G, Prem. M V. Efficient indexing and retrieval of patient information from the big data using MapReduce framework and optimisation. Journal of Information Science, 2023, 49(2): 500-518.

[12] Soltanmohammadi E, Dilek A, Hikmet N. Tailored Partitioning for Healthcare Big Data: A Novel Technique for Efficient Data Management and Hash Retrieval in RDBMS Relational Architectures. Journal of Data Analysis and Information Processing, 2024, 13(1): 46-65.

[13] Kjorveziroski V, Filiposka S. Webassembly as an enabler for next generation serverless computing. Journal of Grid Computing, 2023, 21(3): 34.

[14] Putra S R, Moraes F, Hauff C. Searchx: Empowering collaborative search research. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, 1(1): 1265-1268.

[15] Dai S, Li K, Luo Z, et al. AI-based NLP section discusses the application and effect of bag-of-words models and TF-IDF in NLP tasks. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 2024, 5(1): 13-21.

[16] Yang H J, Choi I Y. Enhancing Search Functionality for Website Posts and Product Reviews: Improving BM25 Ranking Algorithm Performance Using the ResNet-Transformer Model. Journal of The Korea Society of Computer and Information, 2024, 29(11): 67-77.

[17] Dejonckheere F. Automated Microservice Identification in Modular Monolith Architectures. 2024, 101(1): 4.

[18] Putra S R, Grashoff K, Moraes F, et al. On the Development of a Collaborative Search System. DESIRES, 2018, 1(1): 76-82.

[19] Zhou Y J, Yao J, Dou Z C, et al. Dynamicretriever: A pre-trained model-based ir system without an explicit index. Machine Intelligence Research, 2023, 20(2): 276-288.

[20] Paulsen D, Govind Y, Doan A H. Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching. Proceedings of the VLDB Endowment, 2023, 16(6): 1507-1519.

[21] Gomes L, da Silva Torres R, CÃ´rtes M L. BERT-and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: a comparative study. Information and Software Technology, 2023, 160(1): 107217.

[22] Danyal M M, Khan S S, Khan M, et al. Sentiment analysis of movie reviews based on NB approaches using TF-IDF and count vectorizer. Social network analysis and mining, 2024, 14(1): 87.

[23] Kungumaraj E, Lathanayagam E, Saikia U, et al. Neutrosophic Topological Vector Spaces and its Properties. International Journal of Neutrosophic Science (IJNS), 2024, 23(2): 63.

[24] Chung Y, Kraska T, Polyzotis N, et al. Automated data slicing for model validation: A big data-ai integration approach. IEEE Transactions on Knowledge and Data Engineering, 2019, 32(12): 2284-2296.

[25] De Falco A, Caruso F, Su X D, et al. A variational algorithm to detect the clonal copy number substructure of tumors from scRNA-seq data. Nature Communications, 2023, 14(1): 1074.

[26] Zhou X, Shen X, Liu Z, et al. A novel shadow calculation approach based on multithreaded parallel computing. Energy and Buildings, 2024, 312(1): 114237.

[27] Xiao F, Wen J, Pedrycz W, et al. Complex evidence theory for multisource data fusion. Chinese Journal of Information Fusion, 2024, 1(2): 134-159.

[28] Manoharan J S. A novel load balancing aware graph theory based node deployment in wireless sensor networks. Wireless Personal Communications, 2023, 128(2): 1171-1192.

[29] Roberts K, Alam T, Bedrick S, et al. Searching for scientific evidence in a pandemic: An overview of TREC-COVID. Journal of Biomedical Informatics, 2021, 121(1): 103865.

[30] Chen J S, Hersh W R. A comparative analysis of system features used in the TREC-COVID information retrieval challenge. Journal of Biomedical Informatics, 2021, 117(1): 103745.