Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **10**, Number **1**, February **2025**

Data-driven Elasticsearch-based Fast Recommendation of Educational Resources for Students

Mei Xu*

College of Education Sciences
Northwest Normal University, LanZhou 730070, P. R. China
Chongqing Industry Polytechnic College, Chongqing 401120, P. R. China
178263445@qq.com

*Corresponding author: Mei Xu
Received January 25, 2024, revised June 14, 2024, accepted August 11, 2024.

Abstract. *Typically, users need information that is highly relevant to specific verticals, such as e-commerce and education. The data in these verticals are highly structured and closely related to the scenarios. As a result, traditional general-purpose search engines suffer from response delay and recommendation accuracy degradation under high concurrency of large data volume. To solve the above problems, this work proposes a fast recommendation method based on Elasticsearch for vertical search engines in the field of educational resources. First, Elasticsearch is used as a distributed search framework to build a load balancing and clustering model, so as to reduce the response time of the program server in the case of large data throughput. Secondly, the Elasticsearch search engine sorting model is investigated and a TF-IDF integration model incorporating statistical features of users, resources and query item features is proposed. The search relevance of the search engine is reshaped for the education vertical, and the personalised search capability of the search engine is improved. Finally, a hybrid recommendation algorithm incorporating user information is designed by combining a demographic-based recommendation algorithm and a collaborative filtering algorithm based on Alternating Least Squares (ALS). The points are weighted to obtain the user's recommendation list. The proposed method is validated on a dataset with a certain data size. The experimental results show that the TF-IDF integration model can improve the personalised search capability of the search engine to a certain extent. Compared with the single recommendation algorithm, the accuracy of recommendation of the proposed hybrid recommendation method is improved by 3.1% and 10.2% respectively, and the recommendation effect is better.*
Keywords: Elasticsearch; machine learning; recommender systems; vertical search; matrix decomposition; hybrid recommendations

1. **Introduction.** With the increasing amount of data in the background of traditional search engines and the changes in the operation policies of major vendors, people gradually discover various problems in the use of search engines [1, 2]. Most of the traditional search engines we commonly use are built on the basis of syntactic understanding of words and inadequate semantic understanding, so sometimes these problems lead to biased search results [3, 4]. At the same time, traditional search engines also incorporate into the search results the information searched for with various search content irrelevant parameters and information such as bidding rankings, real-time hotspots, etc., which not only reduces the efficiency of the search in terms of accuracy but also greatly reduced.

Rapid recommendation system can provide personalized recommendation of educational resources for college students through efficient full-text search and real-time data analysis functions, thus promoting their personal development and career planning and helping students locate their interests and career direction more accurately.

Vertical search engines are suitable for retrieving information with a clear search intent. Through the use of vertical search engine can accurately and quickly get the information in the relevant field [5]. Vertical search engine is a professional search engine made for a certain industry or a certain thing or event, which is an extension and segmentation made on the basis of traditional search engine [6, 7]. Vertical search engine through the precise screening of the incoming data and the library of data in-depth mining sorting and integration and then pushed to the user, to achieve the optimisation of search results and efficiency.

Intelligent recommendation system as a product of the combination of artificial intelligence technology and modern Internet technology. According to statistics, the information pushed by intelligent recommendation algorithms has occupied more than 50% of the market share in all kinds of network platforms [8]. Intelligent recommendation system can be based on the analysis of various characteristics of resources and users to establish a certain relationship between the two and provide users with the corresponding items or information recommendations [9], to help users quickly find the information they want to obtain, while increasing the user's stickiness to the system. Intelligent recommendation can make full use of explicit and implicit data to analyse the user's thoughts, and can also summarise and summarise the user's characteristics to present the user's information in digital form [10, 11]. The main research content of this work is the accurate search of relevant information in the education vertical, and the intelligent recommendation module is integrated into the search system, trying to display the search results of the search engine more accurately in front of the user.

1.1. **Related Work.** In response to the search engine problem, Google introduced the concept of personalised search in 2004. Google uses the PageRank algorithm to generate ranking results, which initially measures the importance of links, and then combines user data and the search phrases themselves to refine the search results. 2013, Shay Banon created Elasticsearch, a distributed and scalable search engine [12, 13]. Compared to general-purpose search engines such as Google, Elasticsearch can design algorithms based on application scenarios and accepts strictly structured data, making personalised search for vertical domains more flexible.

Vavliakis et al. [14] proposed a personalised search method for e-commerce applications, which takes into account the user's interests, recent browsing behaviour, and the current sales trend of resources to get the best search results. Chen [15] proposed a method to optimise search engine results based on user interaction by generating different search results based on the search topic. Jeong and Kim [16] proposed a personalised search research method based on Elasticsearch, which analyses the user's historical search history and utilisation so that the search engine can interact with the user more intelligently and friendly, and the search results can be generated in a way that the user can interact with the search engine more intelligently and friendly. Intelligent and friendly interaction between the search engine and the user, and the search results are more in line with the actual needs of the user. Kim et al. [17] investigated the personalised search engine keyword extraction methods such as TF-IDF, TextRank and the improved TF-IDF to obtain better search results.

For the intelligent recommendation problem, the recommendation algorithms of most educational resource search engines are still based on collaborative filtering as the core,

which calculates the user's preferences by analysing the user's behavioural data to make recommendations. In practice, most of the educational resource recommendation systems suffer from data sparsity and cold-start problem due to the simplicity of the website's recommendation method and the difficulty of obtaining sufficient user and resource data in a short time. In order to solve the data sparsity problem, Zhou et al. [18] proposed a recommendation system based on the Singular Value Decomposition (SVD) method, which can improve the accuracy of the recommendation results to a certain extent by decomposing and downgrading the data using SVD. Although any matrix can be subjected to SVD dimensionality reduction, the disadvantage is that the complexity of the computation is relatively high and requires a large amount of memory. Considering these problems, Zachariah et al. [19] proposed a matrix decomposition algorithm based on ALS (Alternating Least Squares), which usually decomposes the scoring matrix into two matrices and then fixes the parameters of one matrix to solve for the parameters of the other matrix. By alternatively solving with fixed parameters, the two preference matrices are multiplied as close as possible to the original scoring matrix, so as to make reasonable predictions for the scored data.

In order to solve the data cold start problem, Zhao et al. [20] used a demographic approach to calculate the user similarity using the user's basic attribute information and combined it with a trust mechanism to make recommendations for new users to alleviate the cold start problem. Poongodi et al. [21] proposed a recommendation model based on user trust, and the similarity of user rating and user trust were integrated in the calculation process. For new users who have no behavior records, we recommend them through the trust of users, which effectively alleviates the cold start problem of users.

1.2. **Motivation and contribution.** Considering the complexity and variability of the actual application scenarios, a single algorithm is not well adapted to this changing environment, and has its own shortcomings and deficiencies in the process of use. Therefore, in order to make up for the shortcomings, we can try to combine the advantages of different recommendation methods. In addition, there are some shortcomings in Elasticsearch's raw search functionality when facing the education vertical [22, 23]. For example, when a user performs a search on a search engine, it is particularly important to identify his/her real search intent, rather than providing the user with a thousand and one search results, which puts a demand on the search engine's ability to personalise the search.

Therefore, for vertical search engines in the field of educational resources, this work proposes a fast recommendation method based on Elasticsearch. The main innovations and contributions of this work include:

(1) The Elasticsearch search engine was improved using a distributed cluster routing algorithm to address the problem of reduced programme server performance in the presence of large data throughput.

(2) Aiming at the problem that Elasticsearch's original scoring algorithm only considers word frequency and ignores the user's search intent, this paper proposes a TF-IDF integration model that integrates the statistical features of users and resources and the features of query items. The user's search behaviour not only relies on the scoring results of document relevance, but also includes the statistical features of users and resources, so it can provide users with more personalized search results.

(3) Aiming at the data sparsity and cold-start problem that exists in most of the recommendation systems for educational resources, this paper proposes a hybrid recommendation algorithm incorporating user information by combining a demographic-based recommendation algorithm and a collaborative filtering algorithm based on ALS (alternating least squares).

2. **Elasticsearch search engine based on distributed cluster routing.** The traditional way of building search engine services can slowly become sluggish due to the continuous expansion of data and business, and even in some cases there are scenarios where search fails. The nature of these problems is due to the centralisation of the overall business. The traditional way of centralisation has the advantages of ensuring high integrity of business processes, centralised functionality and ease of management, but at the same time, it will also suffer from various delays and failures brought about by the explosion of data. How to improve the above defects on the basis of the existing vertical search engine has become a key problem that must be solved in the process of building a vertical search engine. In this work, we address the above problems from two different directions: the back-end interface and the data search engine.

2.1. **The back-end interface.** In order to withstand the high throughput and concurrency problems at the interface side, it is necessary to first decompose all the business and functional parts in order to reduce the complexity of the whole programme's functionality. The whole structure of the vertical search engine based on intelligent recommendation is carefully divided according to the business.

Secondly for high concurrency in the overall programme, there are two different approaches to address the problem. (1) Continuously improve the performance bottleneck of the server to reduce the server-side stress caused by data concurrency. (2) Reduce the pressure on individual servers by load balancing across multiple servers. In the first case, if the increase in concurrency is small, it is possible to increase the bottleneck by increasing the configuration of individual servers in order to reduce the pressure on the servers. The user request forwarding load balancing model is shown in Figure 1.

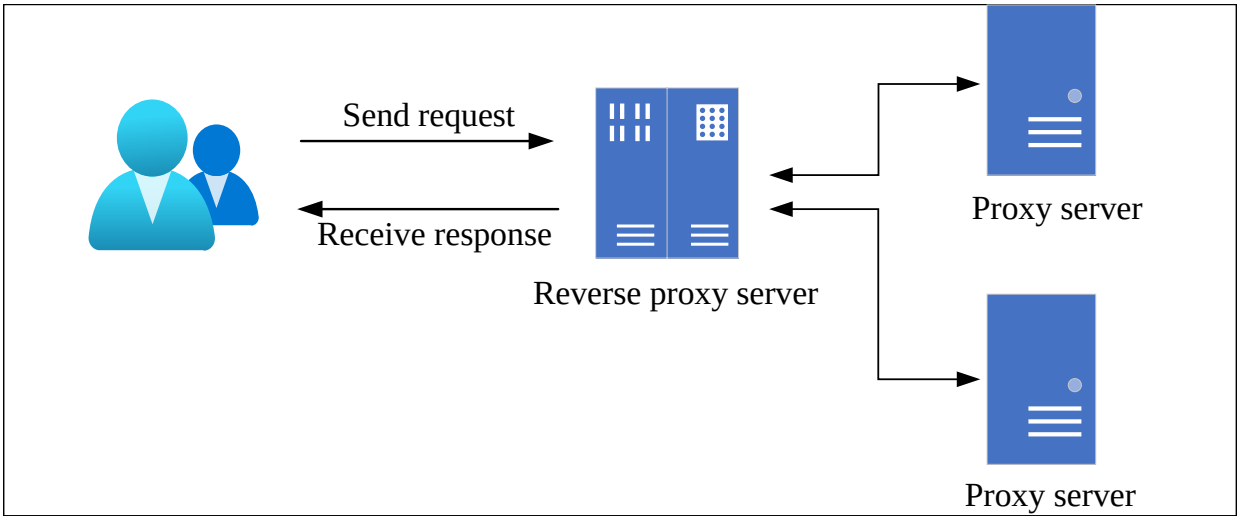

Figure 1. User request forwarding load balancing model

2.2. **Data search engine.** After dealing with the problem of back-end interface pressure, it is still necessary to improve all aspects of the search engine body to ensure that the overall search efficiency and risk resistance. Generally speaking, a single server to set up ElasticSearch and as a search engine body framework in the relatively small business functions of the search volume is not large, the performance of the search engine is acceptable. However, as mentioned in the previous section of the various scenarios, as business continues to expand and concurrency increases, ElasticSearch will also appear their own

bottlenecks, so the need to build a search engine cluster on the basis of the existing single ElasticSearch server.

Through the construction of ElasticSearch cluster we can randomly store the data in different sub-servers, and all the data are reserved for backup information to prevent accidents. In addition, the existence of cluster makes the search engine have the function of multi-server data load balancing, and the appearance of realizing load balancing in the back-end interface program becomes the second protection mechanism to prevent the impact of high concurrency and high throughput on the program, as shown in Figure 2.

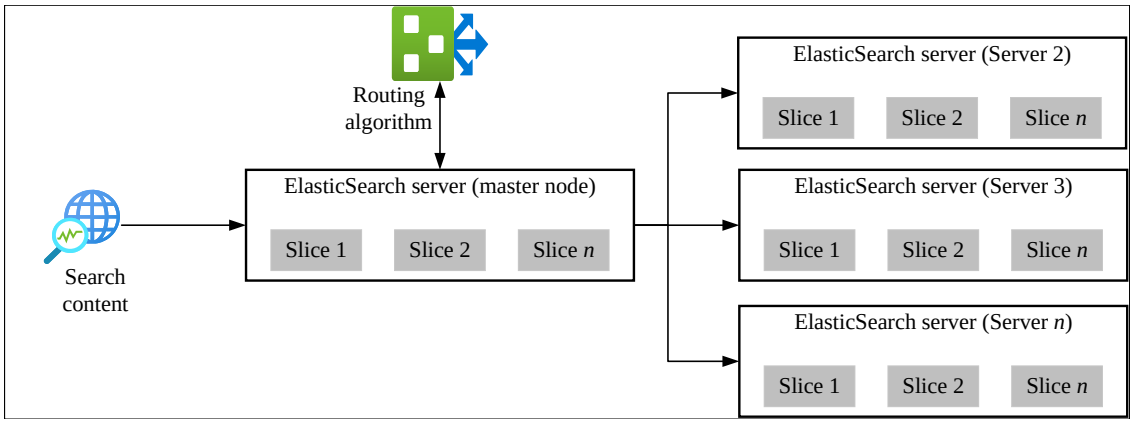

Figure 2. Principles of search engine clustering

Although the use of ElasticSearch clusters can greatly improve the robustness and performance of the overall search engine, due to the fact that the data exists in several different servers, a full search of each slice of each server is required to extract the data when performing a search. There is no direct improvement in search efficiency when compared to a single ElasticSearch server performing the search operation. Therefore, we need to improve the search method on the basis of clustering to improve the efficiency of search.

In this paper, a new routing algorithm is designed in the ElasticSearch data extraction and storage section and this is used as a new routing retrieval algorithm. The time and consumption of searching among multiple servers can be reduced by this routing algorithm as shown below:

$$num = hash(doctype)\%primary\_hards \tag{1}$$

Where *hash* denotes the hash algorithm, *doctype* denotes the type of data text, *primary_shards* denotes the number of primary index slices, and % denotes the modulo operation.

By doing *hash* on *doctype* and modulo *primary_shards*, we get *num* (the location where the data will be stored and retrieved). For example, for the three ElasticSearch cluster servers, "software class" as the file type and hash operation yields a value of 629696930, and the modulo operation on the three servers yields a value of 2. This indicates that this type of data will be deposited into the ElasticSearch cluster servers with a node number of 2 according to the new routing algorithm. ElasticSearch cluster server's slice. Similarly, this routing algorithm can be used to accelerate the overall data reading efficiency by directly targeting the main node slice where the data is stored.

In order to prevent the data from being stored only on a particular server at the time of data storage, the *Quorum* algorithm is required to control the condition of the master

node of the ElasticSearch server at the time of the risk to ensure that a cluster split occurs.

$$Quorum = \text{int}((numberofservice)/2) + 1 \tag{2}$$

where $numberofservice$ indicates the number of servers on the master node.

In ElasticSearch clusters when the number of servers that can be connected is less than the number calculated by $Quorum$, each server can only set this server as the master node, thus preventing the cluster from splitting.

3. **Improvement of the relevance scoring model.**

3.1. **Vector space model.** The search engine calculates the similarity between the query statement and the search results so that the documents that match the user's search criteria are outputted in descending order of relevance. This work focuses on an in-depth study of the Elasticsearch scoring model. For educational search engines, in order to explore the user's real search intent, a weighted hybrid personalised relevance scoring model based on TF-IDF [25] is used on the basis of Lucene's practical scoring function to complete the personalised reordering of the search results and display the optimal list of search results to the user.

Based on the TF-IDF algorithm and the vector space model, the computation of relevance is comprehensively controlled by means of field length normalisation and boosting query weights. The vector space model is a spatial algebraic model that represents a document as a vector composed of feature terms, as shown in Figure 3. In a typical two-dimensional vector space, each vector is represented by the weights of two different words. When t feature terms exist, the two-dimensional space can be extended to t dimensions. In this case, the document and query words are represented by one-dimensional vectors as $d_j = (w_{1,d}, w_{2,d}, \ldots, w_{n,d})$, and the query $q = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$, where $w_{1,d}$ is the document and $w_{1,q}$ the weights of the feature terms in the query.

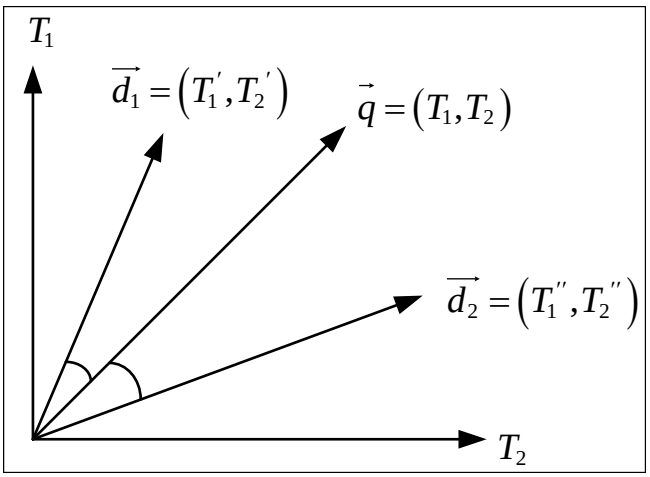

Figure 3. Two-dimensional vector spaces

Based on the document similarity theory, the relevance ranking of documents in keyword search is calculated by comparing the deviation of the angle between each document vector and the original query vector. The cosine similarity is used to measure the similarity of two vectors, which is 1 if the angle between the two vectors is 0°.

$$sim(d_1, q) = \frac{V(d_1) \cdot V(q)}{\|V(d_1)\|\|V(q)\|} = \frac{\sum_{i=1}^{N} w_{i,d} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,d}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}} \tag{3}$$

where $V(d_1) \cdot V(q)$ is the dot product of two vectors with weights, $\|\|$ denotes the Euclidean paradigm, and $w_{i,d}$ denotes the weights of the words appearing in the document d. Based on the vector space model, the relevance of the matching results can be calculated, and the result list can be sorted.

3.2. **Practical scoring functions.** Typically, scoring in Elasticsearch is a process of determining the relevance of retrieved documents based on the user query, word frequency, and some other important parameters. The scoring mechanism in Elasticsearch is inherited from Lucene. For example, the query "relevance and scoring" returns only those documents that match both "relevance" and "scoring". For example, the "relevance and score" query returns only those documents that match both "relevance" and "score". Elasticsearch then applies Lucene's useful scoring functions to score the documents and sort them according to their relevance.

Enter $f(x)$ into the conversion function $G()$.

$$score(q, d) = queryNorm(q) \cdot coord(q, d) \cdot norm(t, d) \tag{4}$$

where d denotes a document, q denotes a query statement, and t denotes a term in the query statement.

$queryNorm(q)$ is a query normalisation factor that enables the comparison of the results of two different queries by normalising the query. The default implementation is shown below:

$$queryNorm = \frac{1}{\sqrt{sumOfSquaredWeights}} \tag{5}$$

where $sumOfSquaredWeights$ is the sum of squares of IDF values for each word in the query.

$coord(q, d)$ evaluates the number of times the query term occurs in a document, providing a higher score for documents that contain more query terms. In a general sense, the more query words appear in a document, the more chances it has to be a good match. However, in some scenarios where synonyms occur, the contribution of this factor will be ignored.

$norm(t, d)$ denotes the field length normalised value. Typically, the shorter the field, the higher its weight. If a word appears in a field like title, it is going to provide a higher relevance contribution than if it appears in a field like content.

$$norm(t, d) = \frac{1}{\sqrt{numTerms}} \tag{6}$$

where $numTerms$ represents the total number of words in the field.

3.3. **Weighted hybrid scoring model based on TF-IDF.** The original scoring algorithm of Elasticsearch search engine only considers the word frequency and ignores the user's search intent. In this paper, we adopt an integrated TF-IDF algorithm model that integrates the statistical features of users and resources and the features of query items, so that the user's search behaviour does not only rely on the scoring results of document relevance, but also includes the statistical features of users and resources.

The features of the query items are mainly derived from the user's search logs, and the semantic relevance between the query and the resource can be obtained by measuring the

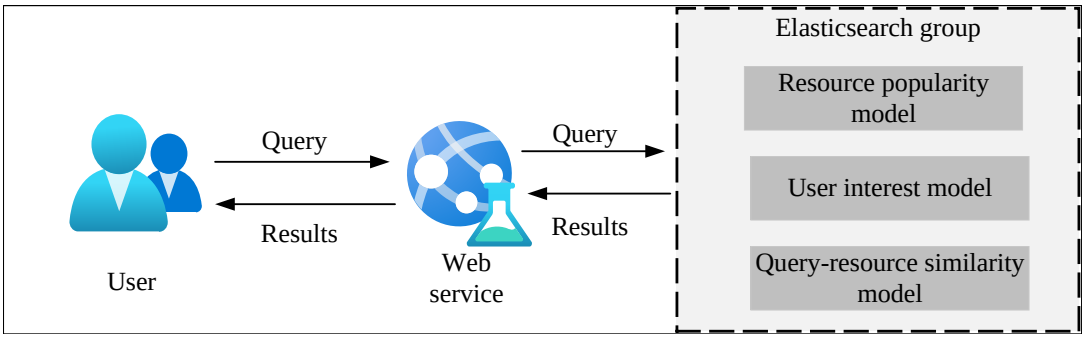

Figure 4. Integrated model architecture

co-occurrence of the query term and the resource. According to the method described in this section, the corresponding semantic relevance between each "query-resource" pair is measured using the TF-IDF algorithm, which models resources and query terms in vector space. In this way, each query term and resource can be described as a vector and matched in the same shared semantic space. In this paper, an integrated algorithmic model is used to remodel the relevance of user query behaviour, as shown in Figure 4.

(1) Resource popularity model. It is more reasonable to improve the relevance score based on the popularity, so by extracting the statistical features of browsing, clicking, and downloading of educational resources, we get a method to calculate the popularity of resources.

$$popularity_i = w_v \frac{views_i}{|views|} + w_c \frac{clicks_i}{|clicks|} + w_t \frac{transactions_i}{|transactions|} \tag{7}$$

where $views_i$, $clicks_i$, $transactions_i$ represent the number of views, clicks, and downloads of resource i, respectively; $|views|$, $|clicks|$, $|transactions|$ denote the total number of views, clicks, and downloads of all resources in the repository, respectively. The contribution of each behaviour to the popularity of a resource is controlled by the w_v, w_c, and w_t parameters.

(2) User interest modelling. Another factor that needs to be taken into account is the user-resource relationship, i.e., the implicit feedback from each user to each resource. The design of this component can tap into the deeper search intent of the user, adding personalised features to the user's search. However, the user's interest will not be static, so the model adds consideration of the time factor. In general, recent interactions with resources are more important than past interactions, and recent user behaviours are more indicative of the user's current interests. Based on this idea, the design of the user interest model is shown below:

$$relevance_{u,i} = w_v \frac{\sum \left(views_{d,u,i} * \left(1 + \frac{1}{1-e^{-x}}\right)\right)}{|views_u|} + w_c \frac{\sum \left(clicks_{d,u,i} * \left(1 + \frac{1}{1-e^{-x}}\right)\right)}{|clicks_u|} + w_t \frac{\sum \left(transactions_{d,u,i} * \left(1 + \frac{1}{1-e^{-x}}\right)\right)}{|transactions_u|} \tag{8}$$

where $views_{d,u,i}$, $clicks_{d,u,i}$, and $transactions_{d,u,i}$ represent the number of views, clicks, and downloads of resource i by user u on the d-th day respectively; $|views_u|$, $|clicks_u|$, and $|transactions_u|$ represent the total number of views, clicks and downloads of resource i by user u; x represents the time interval between the current search behaviour of the user and the d-th day, through which weights can be dynamically assigned to the user's behaviour in history.

(3) Query-resource similarity model. This part is an algorithm based on TF-IDF to calculate similarity, which takes into account the textual similarity of query keywords and resources. The final relevance score depends on the weighted sum of the resource popularity model, user interest model, and query-resource similarity model.

$$recommendationScore_{q,u,i} = w_p \cdot popularity_i + w_r \cdot relevance_{u,i} + w_q \cdot queryScore(q,d) \quad (9)$$

where w_p, w_r, and w_q are the weights of the product popularity model, user interest model, and query relevance model, respectively.

4. **Design of hybrid recommendation algorithms.**

4.1. **Matrix decomposition with alternating least squares.**

For the user cold-start problem existing in the recommender system, a demographic-based recommendation method is used in the offline recommendation module, and the similarity between users is calculated using the attribute vector constructed from their attribute information. For the sparse problem of rating data, for the sparse data problem in the user rating matrix, matrix decomposition and least squares are used to calculate the values of the decomposed matrix, and then the missing values in the user rating matrix are predicted.

The main purpose of using ALS-based collaborative filtering recommendation algorithm in the offline recommendation module is to predict the missing values in the scoring matrix to alleviate the sparsity of the scoring data, which can be roughly divided into two steps: matrix decomposition and alternating least squares. After these two steps, the values of the decomposed matrix are finally calculated, and then they can be multiplied to reduce the original scoring matrix [27] to predict the values that are not in the scoring matrix.

For educational resource recommendation, assuming that there are m users and n educational resources in the educational resource recommendation system, a user rating matrix $R_{m\times n}$ can be constructed based on user ratings as shown in Table 1.

Table 1. Example of a User Rating Matrix

	V_1	V_1	V_3	...	V_n
u_1	1		2	...	5
u_2		5		...	2
u_3	4		4	...	
...		2	4	...	
u_m		3	2	...	3

The user rating matrix $R_{m\times n}$ is firstly matrix-decomposed to obtain two matrices $U_{m\times k}$ and $V'_{k\times n}$, which are then transposed by $V'_{k\times n}$ to obtain $V_{n\times k}$. In order for the decomposed matrix to be as similar as possible to the original matrix after reduction so that it satisfies Equation (10), the values of the decomposed matrix need to be calculated by a specific method. And the determination of these values depends on the assessment of the loss between the predicted score and the true score.

$$R_{m\times n} \approx U_{m\times k} V^T_{n\times k} \quad (10)$$

The calculation is then performed by ALS and optimisation iterations are performed using the value of the loss function. The loss function is usually defined in the manner shown below:

$$Loss(U,V) = \sum_{u\in m, v\in n} \left(r_{uv} - P_u^T q_v\right)^2 \tag{11}$$

Equation (11) becomes equation (12) when the regularisation term is added.

$$Loss(U,V) = \sum_{u\in m, v\in n} \left(r_{uv} - P_u^T q_v\right)^2 + \lambda\left(|P_u|^2 + |q_v|^2\right) \tag{12}$$

When the value of the loss function is smaller represents the decomposition of the matrix is better, multiplication after the reduction is closer to the original matrix, can be approximate instead of the original matrix to analyse the problem. The core principle of ALS-based collaborative filtering recommendation algorithm is to randomly generate the user feature matrix U, fix the value of U. The first order partial derivative of the loss function $Loss(U,V)$ with respect to V is obtained, and the partial derivative is equal to 0. The value of the educational resources feature matrix V is calculated from the derived Equation (13). Then, using the obtained matrix V, the loss function $Loss(U,V)$ is applied to U to find the first-order partial derivative equal to 0. The matrix U is obtained by Equation (14), and these two steps are performed alternately until the maximum number of iterations is exceeded or the algorithm converges and stops.

$$v_a = \left(U_{k\times m} U_{k\times m}^T + \lambda E\right)^{-1} U_{k\times m}^T a \tag{13}$$

$$u_b = \left(V_{k\times n} V_{k\times n}^T + \lambda E\right)^{-1} V_{k\times n}^T b \tag{14}$$

4.2. **User similarity calculation based on user information.** Demographics-based recommendation algorithm is a recommendation method based on basic user information. Assuming that there are m users, users take k basic attributes for calculation, we can construct the user class attribute feature matrix. The $F_1, F_2, \ldots, F_n$ represent the user's basic information. For user u_i, it can be expressed in the form of feature vector according to the user's basic attribute information $u_i = \langle t_{i1}, t_{i2}, \ldots, t_{ik}\rangle$ where t_{ik} denotes the value of the corresponding attribute information of the user.

Generally, through the way of feature processing will be converted to digital values to facilitate the calculation, k represents the number of user features used in the calculation. After processing the feature values, the distance between them can be calculated by the user's feature vector to obtain the user's similarity, in this paper, we use the cosine similarity to calculate the user's neighbour [28], for any user u_i, u_j, their similarity is calculated as follows:

$$sim(u_i,u_j) = \frac{\sum_{s=1}^{k} t_{is}t_{js}}{\sqrt{\sum_{s=1}^{k} t_{is}^2}\sqrt{\sum_{s=1}^{k} t_{js}^2}} \tag{15}$$

For the recommendation of educational resources, ratings reflect the degree of user preference for the resources. Multiple neighbouring users use the average ratings of educational resources to rank the educational resources and then recommend them to users. For a particular educational resource, its average rating R_s is calculated by combining the similarity between users:

$$R_s = \frac{\sum_{i\in U} sim(u_r,u_i) r_{is}}{N_s} \tag{16}$$

where u_r denotes the recommending user, r_{is} denotes the rating of the educational resource s by the user u_i, and N_s denotes the number of times the educational resource s has been rated by the neighbouring users.

4.3. **Hybrid recommendation incorporating user information.** The principle of the proposed hybrid recommendation algorithm is to first find similar users using user information and then use the rating data of similar users predicted by the ALS model to make recommendations.

For each user, the similarity between the user and the recommended user is found based on the user similarity matrix, and then each rating value in the user is multiplied by that similarity to get a new rating value, and in this way all the ratings for each user in the matrix are recalculated.

$$r_{uv} = r_{iv} * sim(u, \bar{u}) \tag{17}$$

where $\bar{u}$ denotes the recommended user, and $sim(u, \bar{u})$ denotes the similarity between the current user and the recommended user.

For a new matrix of user education resource ratings, group the education resources according to their id. For each educational resource, all its ratings are summed and then averaged. Finally, the average rating of each educational resource is calculated using Equation (16) and the final user recommendation list is generated based on the order of the average ratings.

5. Experimental results and analyses.

5.1. **Experimental dataset and environment.** The dataset used in this work is the Edx online course data [29]. The Edx online course dataset is a dataset that contains information about the courses of the Edx online learning platform. The dataset includes information such as the name, description, duration, difficulty level, number of learners, and average rating of the course. This data can be used to analyse aspects such as the popularity of Edx courses, student participation, and course evaluations.

Due to the distributed construction, the system as a whole will be deployed in different servers after different technical architectures and different requirements are divided, and the detailed configuration of each server is shown in Table 2 below. Since the information storage module is built and implemented by the ElasticSearch cluster, each server needs to set up the relevant ElasticSearch framework.

Table 2. Hardware environment

Operating system	**Linux Centos 7.564-bit**	**Linux Centos 7.5 64 bit**	**Linux Centos 7.5 64 bit**
CPU core	2	2	2
Bandwidths	2Mbit/s	2Mbit/s	2Mbit/s
Running memory	4GB	4GB	4GB
Data disc	200G	200G	200G
Intranet IP address	172.17.0.13	172.17.0.28	172.17.0.33

The main body of the vertical search engine is mainly implemented by ElasticSearch cluster, Django back-end API interface program, Scrapy web crawler framework, Nuxt.js front-end framework, Nginx reverse proxy server and other software and frameworks. The software environment of the specific vertical search engine is shown in Table 3 below.

Table 3. Software environment

Jdk	ElasticSearch	Neinx	Django	Nuxtjs	Serapy
15.0.1	7.10.1	1.6.2	3.2	2.14.6	1.6.0

Table 4. Comparison of response time of different search engines

Search term	Elasticsearch/s	Distributed cluster routing based Elasticsearch/ms
English	1.21	400
Data analysis	1.29	286
Computer application	1.73	283
Software development	1.57	244
Communications principle	1.84	318
Mechanical drawing	1.69	289
Mechanical person	1.15	273
...	...	...

5.2. **Distributed Performance Test.** Firstly, we selected 100 different search hot terms as the comparison object for this experiment and compared the traditional Elasticsearch with the Elasticsearch based on distributed cluster routing, as shown in Table 4.

It can be found that the traditional Elasticsearch search engine consumes more time and resources when dealing with individual functions, due to the extremely high coupling of the programmes which increases the total time consumed. On the contrary, the Elasticsearch search engine based on distributed cluster routing successfully decouples the whole business and greatly reduces the probability of encountering risky events.

Second, it was verified that parallel connection of multiple servers can solve the latency problem caused by high concurrent throughput. We used the stress testing tool Postman to simulate multiple concurrent accesses to data respectively. The access response efficiency of the system before and after load balancing was tested. The test simulated 2000 people accessing the program concurrently and the response pairs are shown in Table 5.

Table 5. Comparison of response time of different search engines

	Average response time/ms	Number of successful requests
Elasticsearch	1320	2000
Distributed cluster routing based Elasticsearch	391	2000

The above test data reveals that the Elasticsearch engine based on distributed cluster routing can effectively reduce the corresponding speed of the procedure, and the speed of the engine's search is greatly improved when the new routing algorithm is used in a cluster of multiple servers. A load-balanced pair with 2000 concurrency is shown in Figure 5.

5.3. **Recommendation performance test.** The weight parameters w_p, w_c, and w_t in the weighted hybrid scoring model are 0.127, 0.196, and 0.667, respectively. The results of the multiple tests show that the adopted integrated algorithmic model achieves a relatively superior search quality improvement under the weight assignments of $w_p = 1$, $w_r = 1.2$, and $w_q = 11$.

This paper evaluates the proposed hybrid recommendation algorithms that incorporate user information based on the metrics of accuracy, recall, and F1. The comparison recommendation algorithms include Demography-based single recommendation algorithm, ALS-based collaborative filtering recommendation algorithm, UserCF, ItemCF, and LFM [**?**]. The experiment firstly divides the user rating dataset evenly and randomly into five parts, conducts multiple experiments, and each time only selects one of them as the test

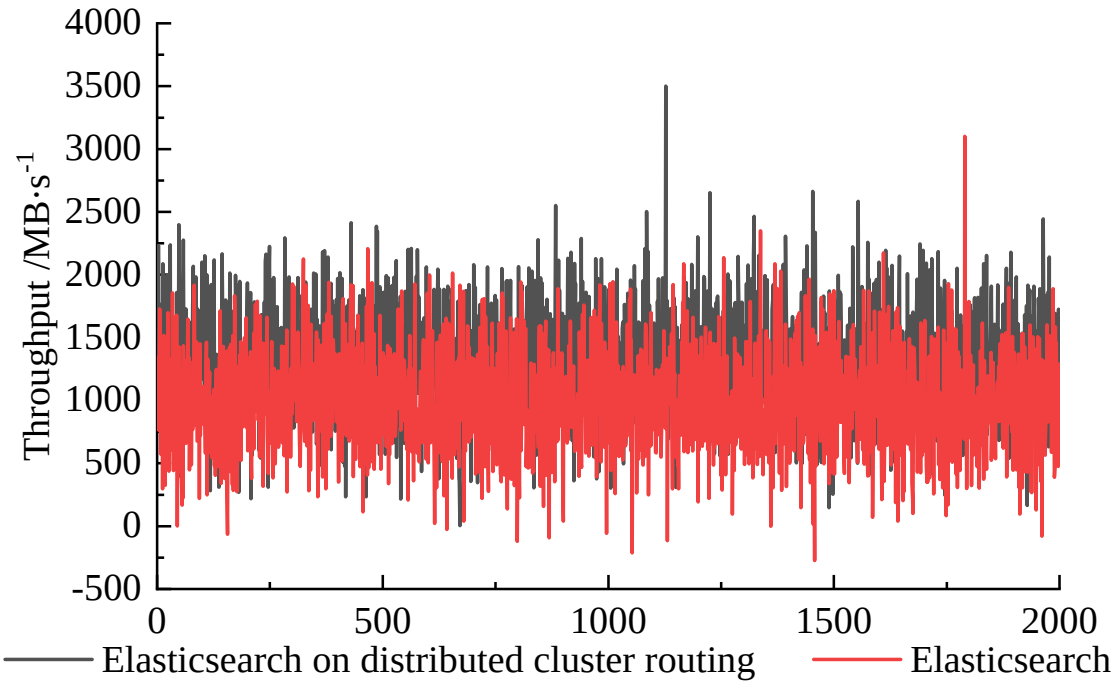

Figure 5. Load Balancing Comparison for 2000 Concurrencies

set, and takes the average of the results measured in multiple experiments for comparison, and the results are shown in Table 6.

Table 6. Comparison of Recommendation Effect under Different Algorithms

Recommended methods	**Accuracy**	**Recall rate**	**F1**
UserCF	0.2113	0.1043	0.1397
ItemCF	0.2087	0.0986	0.1339
LFM	0.2536	0.1142	0.1619
Demography	0.1528	0.0785	0.1037
ALS	0.2246	0.1089	0.1467
Hybrid	0.2653	0.1276	0.1680

It can be seen that the Hybrid approach has the highest recall, accuracy and F1 values of 0.2653, 0.1276 and 0.1680, respectively. This is followed by the LFM approach, which has recall, accuracy and F1 values of 0.2536, 0.1142 and 0.1619. The Demography-based approach performs poorly in all metrics. The new Hybrid recommendation approach incorporates user attributes for computation, which allows for further filtering of the recommendation results. The accuracy of the Hybrid recommendation approach is improved by 3.1% and 10.2% compared to the single recommendation algorithm, respectively.

6. **Conclusion.** This work proposes a fast recommendation method based on Elasticsearch. Firstly, the problem of high throughput and concurrency is addressed from two different directions: the back-end program interface and the data search engine. A new routing algorithm is designed in the ElasticSearch data extraction and storage part and this is used as a new routing retrieval algorithm. A TF-IDF integration model incorporating statistical features of users, resources and query item features is proposed. A hybrid recommendation algorithm incorporating user information is proposed by combining a demographic-based recommendation algorithm and an ALS-based collaborative filtering

algorithm. Experimental results on the Edx online course dataset validate the effectiveness of the proposed method. Although vertical search engines alleviate the problems caused by the impact of big data, how to better improve the performance of search engines under constraints will always be an important improvement direction in the future.

REFERENCES

[1] D. Hawking, N. Craswell, P. Bailey, and K. Griffihs, "Measuring search engine quality," Information Retrieval, vol. 4, pp. 33-59, 2001.
[2] M. Gusenbauer, and N. R. Haddaway, "Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources," Research Synthesis methods, vol. 11, no. 2, pp. 181-217, 2020.
[3] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L.-D. Ibáñez, E. Kacprzak, and P. Groth, "Dataset search: a survey," The VLDB Journal, vol. 29, no. 1, pp. 251-272, 2020.
[4] F. da Veiga Leprevost, S. E. Haynes, D. M. Avtonomov, H.-Y. Chang, A. K. Shanmugam, D. Mellacheruvu, A. T. Kong, and A. I. Nesvizhskii, "Philosopher: a versatile toolkit for shotgun proteomics data analysis," Nature Methods, vol. 17, no. 9, pp. 869-870, 2020.
[5] R. Tso, K. Huang, Y.-C. Chen, S. M. M. Rahman, and T.-Y. Wu, "Generic Construction of Dual-Server Public Key Encryption with Keyword Search on Cloud Computing," IEEE Access, vol. 8, pp. 152551-152564, 2020.
[6] T.-Y. Wu, C.-M. Chen, K.-H. Wang, and J. M.-T. Wu, "Security Analysis and Enhancement of a Certificateless Searchable Public Key Encryption Scheme for IIoT Environments," IEEE Access, vol. 7, pp. 49232-49239, 2019.
[7] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," Journal of the Chinese Institute of Engineers, vol. 42, no. 1, pp. 20-28, 2019.
[8] Y.-T. Chang, H.-R. Yang, and C.-M. Chen, "Analysis on improving the application of machine learning in product development," The Journal of Supercomputing, vol. 78, no. 10, pp. 12435-12460, 2022.
[9] Q. Zhang, J. Lu, and Y. Jin, "Artificial intelligence in recommender systems," Complex & Intelligent Systems, vol. 7, pp. 439-457, 2021.
[10] B. Cao, Y. Zhang, J. Zhao, X. Liu, Ł. Skonieczny, and Z. Lv, "Recommendation based on large-scale many-objective optimization for the intelligent internet of things system," IEEE Internet of Things Journal, vol. 9, no. 16, pp. 15030-15038, 2021.
[11] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo, "A review of content-based and context-based recommendation systems," International Journal of Emerging Technologies in Learning, vol. 16, no. 3, pp. 274-306, 2021.
[12] R. A. Hamid, A. S. Albahri, J. K. Alwan, Z. Al-Qaysi, O. S. Albahri, A. Zaidan, A. Alnoor, A. H. Alamoodi, and B. Zaidan, "How smart is e-tourism? A systematic review of smart tourism recommendation system applying data management," Computer Science Review, vol. 39, 100337, 2021.
[13] M. S. Divya, and S. K. Goyal, "An advanced and quick search technique to handle voluminous data," Compusoft, vol. 2, pp. 171-175, 2013.
[14] K. N. Vavliakis, G. Katsikopoulos, and A. L. Symeonidis, "E-commerce Personalization with Elasticsearch," Procedia Computer Science, vol. 151, pp. 1128-1133, 2019.
[15] L.-C. Chen, "A study of optimizing search engine results through user interaction," IEEE Access, vol. 8, pp. 79024-79045, 2020.
[16] J.-H. Jeong, and C.-J. Kim, "A Dynamic Recommendation Architecture and Procedure Based on Elasticsearch," Journal of Knowledge Information Technology and Systems, vol. 15, no. 4, pp. 463-472, 2020.
[17] J.-Y. Kim, K.-H. Jo, J. Park, S.-H. Jung, and C.-B. Sim, "A Development of Optimal Travel Course Recommendation System based on Altered TSP and Elasticsearch Algorithm," Journal of Korea Multimedia Society, vol. 22, no. 9, pp. 1108-1121, 2019.
[18] X. Zhou, J. He, G. Huang, and Y. Zhang, "SVD-based incremental approaches for recommender systems," Journal of Computer and System Sciences, vol. 81, no. 4, pp. 717-733, 2015.
[19] D. Zachariah, M. Sundin, M. Jansson, and S. Chatterjee, "Alternating least-squares for low-rank matrix reconstruction," IEEE Signal Processing Letters, vol. 19, no. 4, pp. 231-234, 2012.

[20] W. X. Zhao, S. Li, Y. He, L. Wang, J.-R. Wen, and X. Li, "Exploring demographic information in social media for product recommendation," Knowledge and Information Systems, vol. 49, pp. 61-89, 2016.
[21] M. Poongodi, V. Vijayakumar, B. Rawal, V. Bhardwaj, T. Agarwal, A. Jain, L. Ramanathan, and V. Sriram, "Recommendation model based on trust relations & user credibility," Journal of Intelligent & Fuzzy Systems, vol. 36, no. 5, pp. 4057-4064, 2019.
[22] M. Kim, and D. Kim, "A Suggestion on the LDA-Based Topic Modeling Technique Based on ElasticSearch for Indexing Academic Research Results," Applied Sciences, vol. 12, no. 6, 3118, 2022.
[23] J. R. Andersson, J. A. Moya, and U. Schwickerath, "Anomaly Detection for the Centralised Elasticsearch Service at CERN," Frontiers in big Data, vol. 4, 718879, 2021.
[24] H. Haugerud, M. Sobhie, and A. Yazidi, "Tuning of Elasticsearch Configuration: Parameter Optimization Through Simultaneous Perturbation Stochastic Approximation," Frontiers in big Data, vol. 5, 686416, 2022.
[25] L. Zhang, "Research on case reasoning method based on TF-IDF," International Journal of System Assurance Engineering and Management, vol. 12, pp. 608-615, 2021.
[26] A. B. Ayed, I. Biskri, and J.-G. Meunier, "An End-to-End Efficient Lucene-Based Framework of Document/Information Retrieval," International Journal of Information Retrieval Research, vol. 12, no. 1, pp. 1-14, 2022.
[27] S. Huang, Z. Kang, and Z. Xu, "Auto-weighted multi-view clustering via deep matrix decomposition," Pattern Recognition, vol. 97, 107015, 2020.
[28] H. Zhang, L. Liu, H. Zhou, H. Sun, and N. Zheng, "CMD: controllable matrix decomposition with global optimization for deep neural network compression," Machine Learning, vol. 111, no. 3, pp. 831-851, 2022.
[29] J. Zienkowski, and F. Lambotte, "Agency as an emerging phenomenon in the construction of massive open online courses: a discursive–material approach to the techno-pedagogical edX platform and its forums," Learning, Media and Technology, pp. 1-14, 2023.
[30] R. Shen, "A Recommender System Integrating Long Short-Term Memory and Latent Factor," Arabian Journal for Science and Engineering, vol. 47, no. 8, pp. 9931-9941, 2022.