

DESIGN AND IMPLEMENTATION OF AN INTELLIGENT RECOMMENDATION SYSTEM FOR COLLEGE ENTRANCE EXAMINATION APPLICATION PREFERENCES

HongFu Zeng, ZhaoMin Liang*, FanRui Wei, YiXun Lu
College of Artificial Intelligence, Nanning University, Nanning 530007, Guangxi, China.
Corresponding Author: ZhaoMin Liang, Email: minzaa2000@qq.com

Abstract: With the increasing societal emphasis on educational equity in recent years and the continuous rise in the number of applicants for the National College Entrance Examination (NCEE), intelligent and accurate information-based reference platforms for college major selection and application processes have become crucial. Empirical evidence indicates that higher education program matching systems, which leverage complex data analysis of historical admissions data through an information mining architecture, have contributed to more scientific and rational resource allocation in educational institutions, while also enhancing fairness and improvement within the educational ecosystem. However, existing service solutions still exhibit shortcomings that need to be addressed. Concretely, this project adopts a Browser/Server (B/S) architecture during its design and development phases, utilizing technology that separates data presentation from business logic. The primary integrated development environments (IDEs) employed include mainstream tools such as IntelliJ IDEA, PyCharm, and Visual Studio Code (VSCode). The platform incorporates functional modules such as user authentication, student registration workflows, institutional information query operations, simulated application submissions, and an institution recommendation mechanism based on candidate behavior. This modular functional design inherently improves the overall user experience.

Keywords: NCEE application preferences submission; Collaborative filtering algorithm; B/S architecture

1 INTRODUCTION

With the continuous development of China's economy, society has placed new demands on talent cultivation. To adapt to these changes, national and local education authorities have introduced a series of policies and measures to reform the National College Entrance Examination (NCEE). To uphold the principle of fairness in college application processes, many provinces in China have implemented a "discipline-oriented parallel application model," commonly referred to as the "First-Choice Discipline Priority" system. This model prioritizes applicants' academic interests and fundamentally transforms the traditional "university + major" application approach. The "discipline + university" parallel model significantly reduces instances of "high scores leading to low-tier admissions," improves score utilization efficiency, and enhances fairness for the majority of candidates. Another highlight of the NCEE reform is the adoption of a "3+3 subject combination system," which breaks the traditional arts-science division and allows students to select subjects based on their interests and academic strengths, thereby granting them greater autonomy in decision-making. Currently, numerous platforms and software tools exist to assist with NCEE application processes[1]. However, most of these tools underperform in areas such as real-time content updates, scientific rigor of recommendation models, and overall user-friendliness. Field research reveals persistent issues, including delays in dynamic information synchronization, oversimplified foundational algorithmic models, and suboptimal user interface design. To address these challenges, this project developed an intelligent recommendation system for college applications based on a Browser/Server (B/S) architecture with distinct frontend and backend layers. The solution leverages a diversified and flexible technology stack alongside tailored recommendation algorithms. By emphasizing a robust system architecture, students and parents can intuitively access more convenient, personalized assistance and enhanced information security. Additionally, critical performance metrics—such as recommendation accuracy and operational workflow efficiency—have been optimized to ensure a seamless user experience.

2 RECOMMENDATION ALGORITHM FOR HIGHER EDUCATION INSTITUTIONS BASED ON USER BEHAVIOR

2.1 User-Based Collaborative Filtering Algorithm

The system employs user-based collaborative filtering as its core recommendation algorithm[2]. The implementation process comprises the following key steps:

First, multi-dimensional behavioral data including user browsing histories and rating records are collected to construct a user-item interaction matrix. To address data sparsity issues, appropriate similarity metrics such as Pearson correlation coefficient or cosine similarity are selected for computation. Subsequently, the algorithm identifies K-nearest neighbors with preference patterns most aligned to the target user from the massive user pool. Finally, by analyzing the interest

preferences of these similar users, potential items of interest for the target user are predicted through weighted calculations, thereby achieving personalized recommendations. As illustrated in Figure 1:

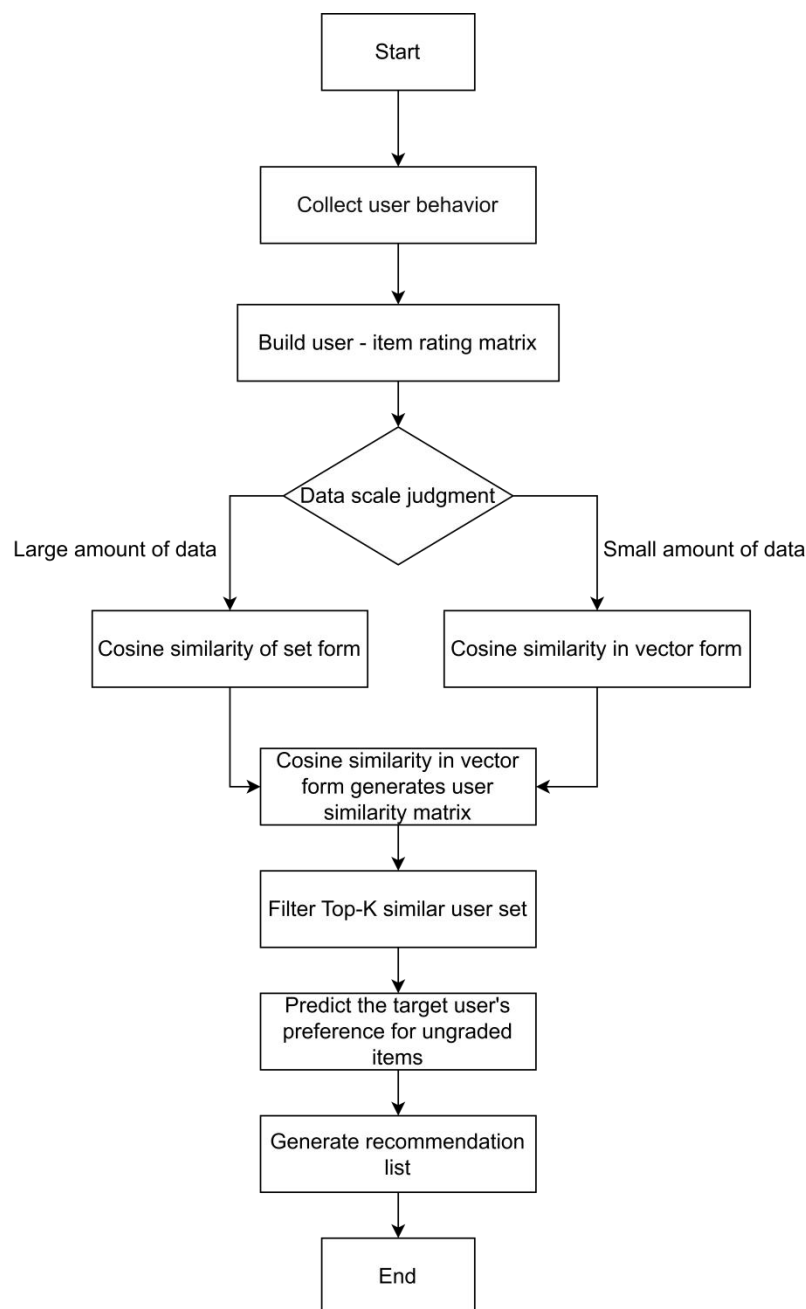


Figure 1 Recommendation Algorithm Construction Process Analysis

The main difference between the formulas for cosine similarity and Jaccard coefficient lies in the handling of the denominator terms: cosine similarity is the product of the number of items, while Jaccard coefficient is the union of user-item pairs, as shown in Formula (1).

$$\text{sim}(A, B) = \frac{|N(A) \cap N(B)|}{\sqrt{|N(A)| * |N(B)|}} \quad (1)$$

with the following definitions:

$\text{sim}(A, B)$: Similarity between users A and B.

$N(A)$: Set of items selected by user A.

$N(B)$: Set of items selected by user B.

$|N(A) \cap N(B)|$: Number of institutions co-selected by users A and B.

$\sqrt{|N(A)| * |N(B)|}$: The geometric mean of the number of higher education institutions selected by User A and User B.

Since conventional data is usually presented in a structured two-dimensional table (with both rows and columns being vectors), similarity measurement is generally calculated using vector formulas. However, it should be noted that this vector-based measurement method has certain limitations and is not applicable to the comparison of set-type data, as shown in Formula (2).

$$\text{sim}(A, B) = \cos(A, B) = \frac{A \cdot B}{|A| \cdot |B|} \quad (2)$$

with the following definitions:

$\text{sim}(A, B)$ Indicates the degree of similarity between User A and User B.

$|A|$: Similarity between users A and B.

$|B|$: Norm (magnitude) of user A's behavior vector.

$A \cdot B$: Dot product of behavior vectors, quantifying directional alignment.

$|A| \cdot |B|$: Product of vector norms for normalization (mitigating rating scale disparities).

In practical applications, data sets are usually of large scale, resulting in a highly sparse data matrix and a large number of zero values in the feature vectors. For different data scales, corresponding measurement methods need to be adopted: large-scale sparse data is suitable for similarity calculation based on sets, while small-scale data can use vector space measurement.

2.2 Implementation Steps of Collaborative Filtering Based on Users

2.2.1 Data preparation stage

Suppose there are four students (Student A, Student B, Student C, and Student D) filling out their college entrance examination applications. The universities they apply for are shown in Table 1:

Table 1 Statistics Table Filled in by Universities

Student	Fill in the college application form
Student A	Tsinghua University, Peking University
Student B	Tsinghua University, Fudan University
Student C	Peking University, Shanghai Jiao Tong University
Student D	Fudan University, Zhejiang University

In the recommendation of college entrance examination applications: userRatings indicates students' preferences for choosing colleges. itemUsers records which students have selected each institution. The key code is as follows:

```
private Map<String, Map<String, Double>> userRatings;
```

```
private Map<String, List<String>> itemUsers;
```

```
private Map<String, Integer> userIndex;
```

```
private Map<Integer, String> indexUser;
```

```
private Long[][] sparseMatrix;
```

2.2.2 Construct an inverted list of institutions and students

The inverted list of institutions and students records which students have applied to each institution. Based on the above data, the inverted list of institutions and students constructed is shown in Table 2 as follows:

Table 2 Inverted list of Institutions and Students

Colleges and universities	Fill in the student form
Tsinghua University	Student A, Student B
Peking University	Student A, Student C
Fudan University	Student B, Student D
Shanghai Jiao Tong University	Student C
Zhejiang University	Student D

2.2.3 Establish the user sparse matrix

The user sparse matrix is used to calculate the similarity between users. The matrix elements ($M_{\{i,j\}}$) represent the number of colleges and universities filled in jointly by User i and user j .

Let the indices of student A, student B, student C and student D be 0, 1, 2 and 3 respectively. Based on the item-user inverted table, the calculated user sparse matrix is shown in Figure 2:

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 2 User Sparse Matrix

Here, $M_{0,1}=1$ indicates that student A and student B jointly applied to one university (Tsinghua University); $M_{0,2}=1$ indicates that student A and student C jointly applied to one university (Peking University).

2.2.4 Similarity calculation

The cosine similarity formula is used to calculate the similarity between users, as shown in Formula 3 [3]:

$$\text{sim}(u, v) = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| * |N(v)|}} \quad (3)$$

Here, u and v denote two distinct students. $N(u)$ and $N(v)$ represent the sets of universities to which students u and v have applied, respectively. $|N(u) \cap N(v)|$ denotes the number of universities that both students u and v have applied to, while $|N(u)|$ and $|N(v)|$ represent the total numbers of universities applied to by students u and v , respectively.

The core code is as follows:

```
Integer id1 = this.userIndex.get(user1);
Integer id2 = this.userIndex.get(user2);
if(id1==null || id2==null) return 0.0;
return this.sparseMatrix[id1][id2]/Math.sqrt(userRatings.get(indexUser.get(id1)).
size()*userRatings.get(indexUser.get(id2)).size());
Calculate the similarity between student A and other students:
sim(A,B)=1/√(2*2)=0.5;
sim(A,C)=1/√(2*2)=0.5;
sim(A,D)=0/√(2*2)=0
```

Select similar neighbors: A and B

Obtain the institutions selected by neighbors but not applied to by A: Among the universities applied to by student B, the university that student A did not apply to is "Fudan University". Among the universities applied to by student C, the university that student A did not apply to is "Shanghai Jiao Tong University".

Generate recommendation results: Fudan University, Shanghai Jiao Tong University.

2.2.5 Recommendation generation

(1) To compute the similarity between the target student and all other students, one should first iterate through all student user data. Then, utilize the pre - defined cosine similarity algorithm to calculate the similarity of the college application preferences between the target student and each of the other students. This procedure excludes the target student itself to guarantee the objectivity of the calculation results. The results are stored in a key - value pair set, where the key represents the user ID and the value represents the similarity score.

(2) Select the top K neighbor students with the highest similarity. Sort all the calculated similarity results in descending order and pick the top K users with the highest similarity as "neighbors". Here, the number of neighbors is controlled by setting the numRecommendations parameter. This not only ensures the diversity of recommendations but also avoids the introduction of noisy data. A dynamic judgment mechanism is employed. When the actual number of similar users is less than K, an automatic adjustment will be made.

(4) Aggregate the colleges chosen by the neighbors but not filled in by the target student. Traverse each neighbor's college application record, collect all the colleges that the target student has not filled in, and record the scores of these colleges in the neighbor users.

(5) Sort and recommend the top N colleges by score. The system sorts the aggregated candidate colleges in descending order of score using the Java Stream API to achieve efficient sorting operations, and uses the limit method to extract the top N colleges with the highest scores.

3 CONSTRUCTION OF INTELLIGENT QUESTION-ANSWERING FUNCTIONS BASED ON LARGE MODELS

The emergence of AI - based intelligent question - answering systems stems from the rapid progress in the field of artificial intelligence. In particular, breakthroughs in natural language processing and deep learning technologies have empowered machines to comprehend and generate human language with greater precision.

Simultaneously, the maturation of big data and cloud computing technologies has laid a foundation for the storage, analysis, and real - time processing of vast amounts of educational data. With the progressive advancement of college entrance examination reforms and the escalating complexity of college application processes, traditional consultation approaches have proven inadequate in meeting the individualized requirements of examinees.

The widespread adoption of mobile internet has also transformed the way people access information, thereby propelling the application of intelligent question - answering systems within the educational domain. Moreover, the endorsement of national education informatization policies and the booming development of the education technology market have further expedited the implementation and refinement of AI - based intelligent question - answering in educational scenarios such as college entrance examination application.

In the intelligent college entrance examination application recommendation system, AI - powered intelligent question - answering offers efficient and accurate application support to examinees via natural language interaction and big data analysis. It can promptly address queries and provide risk alerts, such as "aspirational, safe, and conservative" strategies, thus assisting examinees in making informed decisions and optimizing their application selections.

3.1 Analysis of the AI Intelligent Question-Answering Process

Analysis of the AI intelligent question-answering process: The essence of AI intelligent question answering is a process from user questions to machine responses[4]. As shown in Figure 3:

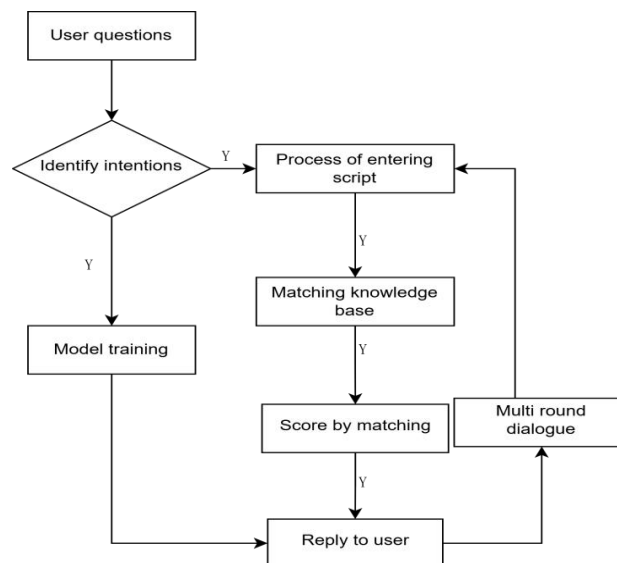


Figure 3 Analysis of the AI Intelligent Question-Answering Process

3.2 Implementation of AI-based Intelligent Question-Answering

In the intelligent college entrance examination recommendation system, the AI - enabled intelligent question - answering feature leverages the DeepSeek model, which has gained significant popularity in recent times. Specifically, this system makes use of the DeepSeek - V3 model to facilitate AI - based dialogues. The DeepSeek - V3 model can be invoked by specifying "model: deepseek - chat". The AI intelligent question - answering mechanism is designed to aid examinees in making choices and submitting applications to their preferred institutions of higher education.

3.2.1 Principles of the DeepSeek-V3 model

The core architecture of DeepSeek-V3 is based on the Transformer model[5], which is a neural network architecture entirely based on the attention mechanism. The Transformer model consists of an encoder and a decoder, and DeepSeek-V3 may choose to use the encoder, the decoder, or a combination of both depending on the task requirements. The model diagram is shown in Figure 4.

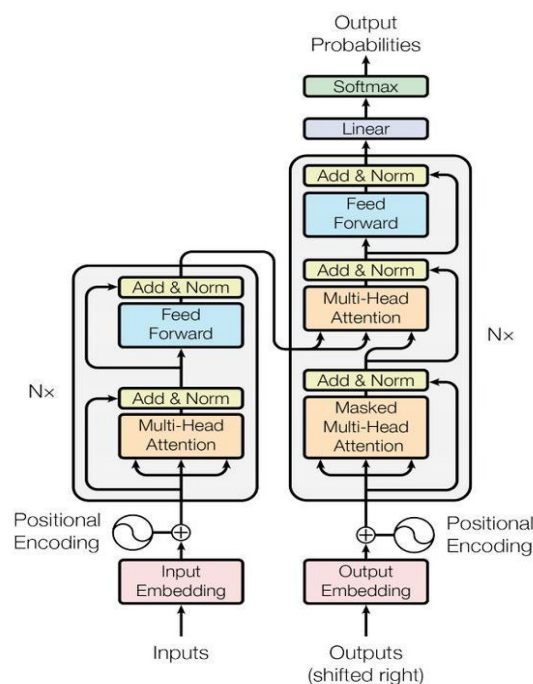


Figure 4 Diagram of the Transformer Model

The following are the core components of the Transformer.

(1) Self - attention Mechanism

This mechanism transcends the limitations of traditional sequence processing, which is constrained to sequential reading. Instead, it enables parallel analysis of the relationships among every word unit within a sequence. By leveraging the dynamic relationships between words, it computes the weights of various possible associations among vocabulary in the current scenario. Moreover, it autonomously evaluates the significance of each context, facilitating the extraction of global semantic features.

(2) Multi - head Attention

Operating through a parallel subspace framework, multi - head attention projects the input features into distinct representational spaces for independent attention computations. Subsequently, the semantic features from these subspaces are integrated. This architectural design effectively captures diverse connections among words from multiple perspectives, enriching the model's understanding of semantic relationships.

(3) Feed - forward Neural Network

Following each attention processing layer, a fully - connected feed - forward neural network is employed. Its primary functions include performing non - linear feature transformations and increasing the dimensionality of features, thereby enhancing the model's representational capacity. This process allows the model to better capture and represent complex patterns and information within the data.

(4) Positional Encoding

In order to resolve the problem that the model is unable to process temporal sequences in an explicit manner, an embedding representation containing positional information is incorporated. The positional encoding is added to the word vectors and subsequently presented to the model as input parameters. In this way, the model can establish an ordered relationship, as depicted in Figure 5.

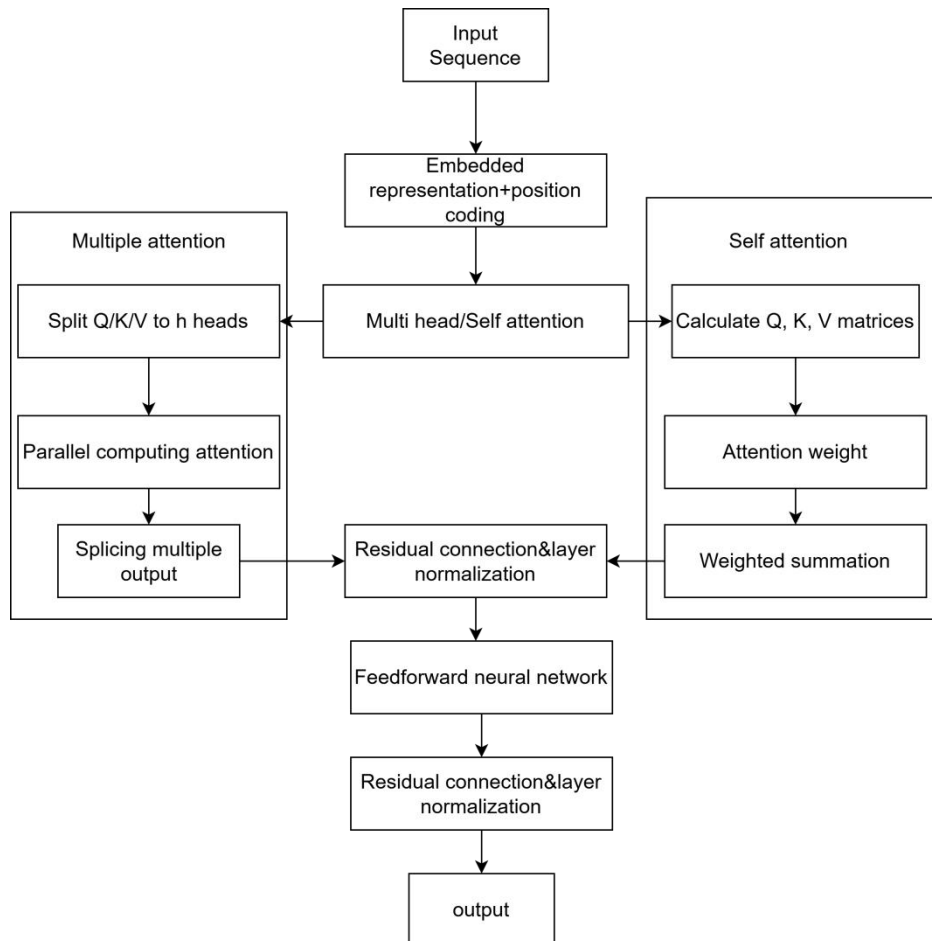


Figure 5 Diagram of the Transformer Model

3.2.2 Implement the interface for AI intelligent question-answering functionality

Apply for an API key on the DeepSeek official website as shown in Figure 5.

API keys

All your APIkeys are in the list. The APIkeys can only be copied when they are created. Please save them properly. Do not share your API key with others, or expose it in the browser or other client code. In order to protect the security of your account, we may automatically disable the APIkey that we found has been disclosed. We did not track the usage of APIkey created before April 25, 2024.


name	Key	Creation date	Latest use date	
zhf	sk-86aef*****86f5	March 8, 2025	April 19, 2025	 

Figure 5 Apply for API Key

- (1) After creating the API key, you can start building the SpringBoot project. Based on the SpringBoot 3x version, set up a project.
- (2) Incorporate the dependency of 'spring-ai-openai-spring-boot-starter'. Spring AI offers Spring Boot auto - wiring functionality for the OpenAI Chat Client. Make adjustments to the configuration file (application.yml).

4 SYSTEM DESIGN AND IMPLEMENTATION

4.1 System Architecture Design

- (1) The intelligent recommendation system of college entrance examination preference adopts the MVC framework to build the system architecture. MVC is divided into three basic parts: Model, View and Controller [6], as shown in Figure 6.

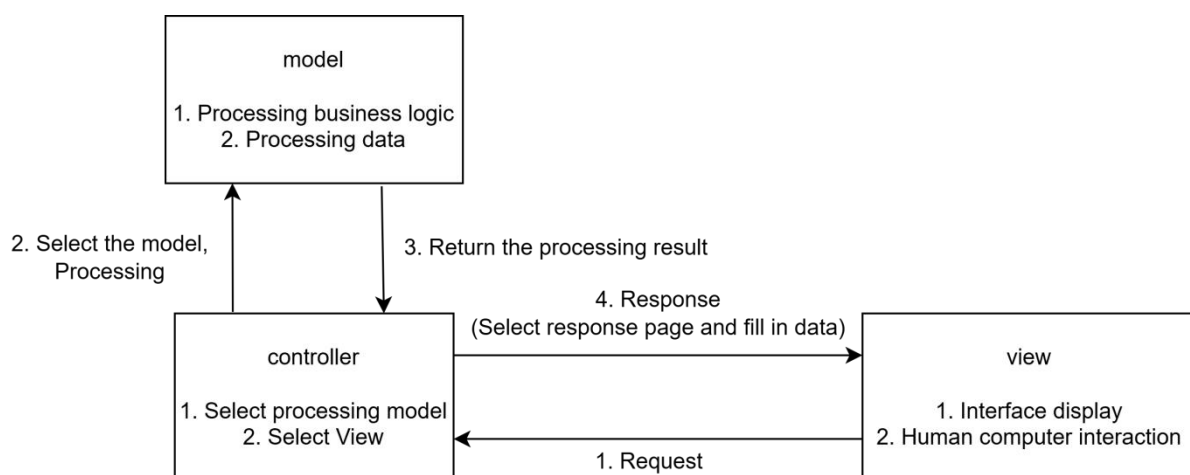


Figure 6 MVC Architecture

- (2) The view component is dedicated to interface presentation and user interaction. It realizes the functions of information display and operation response through carriers such as web pages and forms, as depicted in Figure 7.

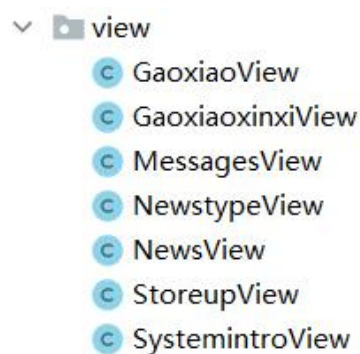


Figure 7 View Layer

- (3) As the core hub of the MVC architecture, the controller is mainly responsible for request routing and process scheduling. Its core functions include: parsing client requests, invoking business models to process data, and determining the view jump path. As shown in Figure 8.

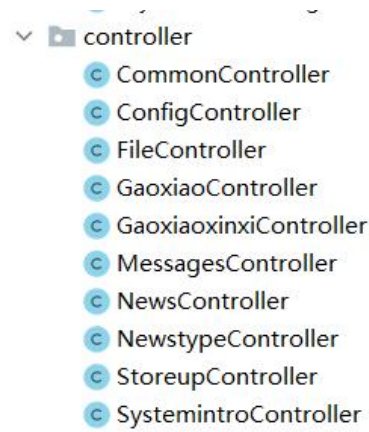


Figure 8 Controller Layer

(4) As the carrier of business logic, the model encapsulates core data structures and processing rules. It is responsible for receiving requests submitted by the view layer, executing the established operation process, and feeding back the processing results to the presentation layer, as shown in Figure 9.

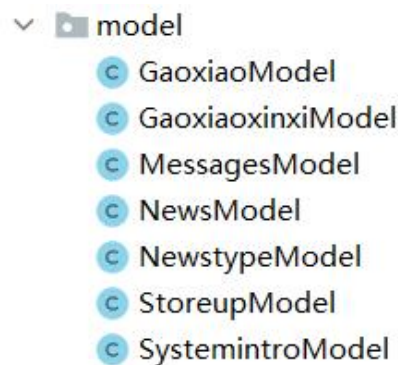


Figure 9 Model Layer

4.2 System Function Module Design

The main functions of the intelligent recommendation system for college entrance examination applications can be divided into the front-end page and the back-end management. Users can register and log in to the system according to their needs[7], browse information such as the addresses, scores, and rankings of universities[8], and the system also recommends universities that may be of interest to each user. The management end mainly maintains user and other data. The system's functional structure is shown in Figure 10.

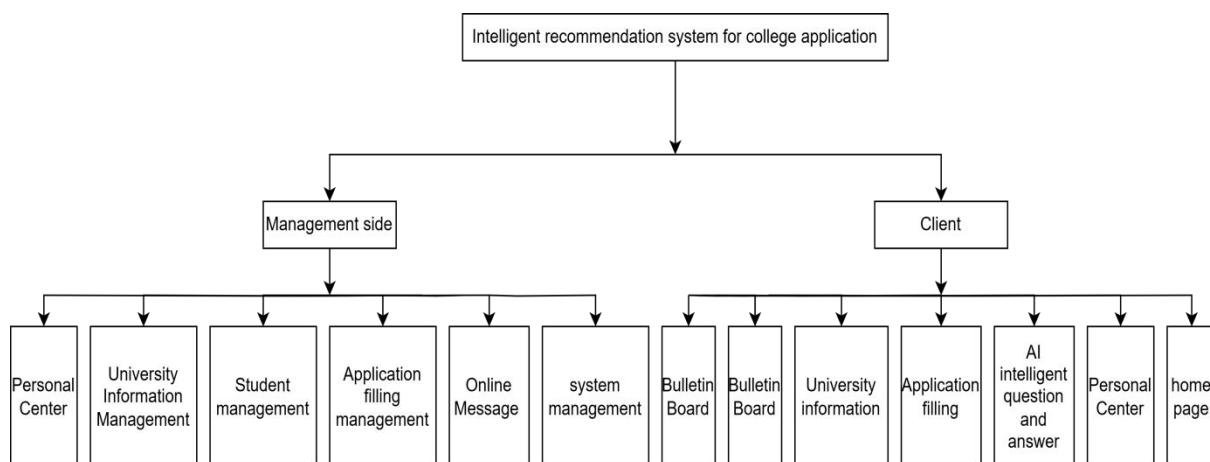


Figure 10 System Function Structure Diagram

4.3 Database Design

4.3.1 ER diagram design

The ER diagram (Entity-Relationship diagram) is a core tool in database design, used to visually present the attributes of key entities (such as users, institutions, majors, etc.) in the system and their interrelationships. It forms the main database E-R relationship diagram of the intelligent college entrance examination volunteer recommendation system, as shown in Figure 11.

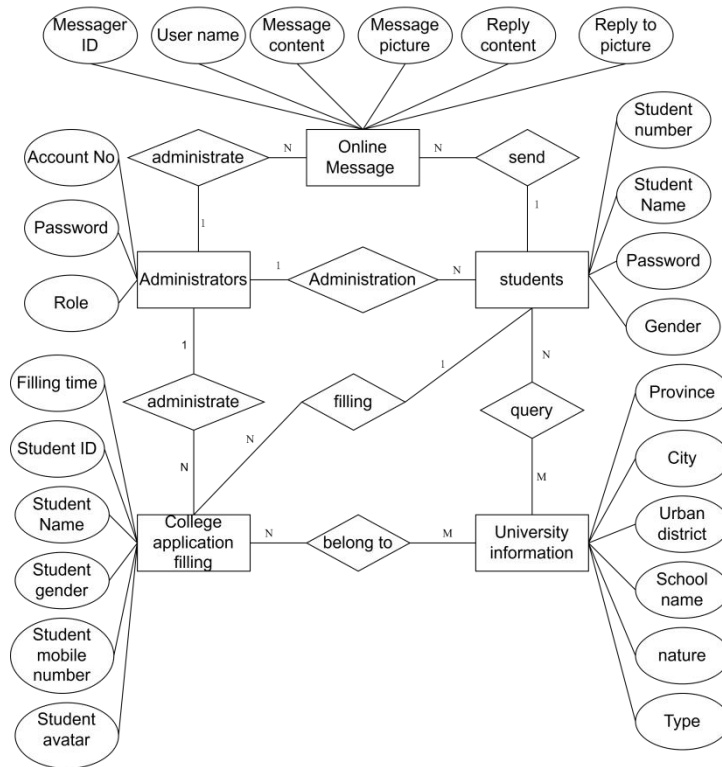


Figure 11 E-R Diagram of the Intelligent Recommendation System for College Entrance Examination Voluntary Applications

4.3.2 Design of data tables

(1) The Online Message Table is used to record users' inquiries and administrators' responses. It encompasses the message content, image attachments, and reply information, as presented in Table 3.

Table 3 Online Message Board

Field Name	Data Type	Length	Field Explanation	Primary Key	Default Value
id	bigint		Primary Key	Primary Key	
addtime	timestamp		Creation Timestamp		CURRENT_TIMESTAMP
userid	bigint		Commenter ID		
username	varchar	200	User Name		
avatarurl	longtext	4294967295	Profile Picture		
content	longtext	4294967295	Message Content		
cpicture	longtext	4294967295	Message Image		
reply	longtext	4294967295	Reply Content		
rpicture	longtext	4294967295	Reply Image		

(2) The Higher Education Institution Information Table stores the enrollment data of colleges and universities across the country. It encompasses key indicators such as region, institution nature, academic discipline type, admission scores, and rank positions. As presented in Table 4.

Table 4 Information Table of Higher Education Institutions

Field Name	Data Type	Length	Field Explanation	Primary Key	Default Value
id	bigint		Primary key	Primary key	
addtime	timestamp		Creation time		CURRENT_TIMES TAMP
sheng	varchar	200	Province		
shi	varchar	200	city		
qu	varchar	200	District		
xuexiaoming	varchar	200	School Name		
xingzhi	varchar	200	Nature		
leixing	varchar	200	Type		

zuidifen	double		The lowest score
zdwc	varchar	200	The lowest rank
biaoqian	varchar	200	Label
pici	varchar	200	Batch
xueke	varchar	200	Subject
zhuanYe	varchar	200	professional

(3) The application form for college admission records the information of the candidates' applications, including the selection of colleges and majors, as well as the time of application, etc. As shown in Table 5.

Table 5 College Entrance Examination Voluntary Application Form

Field Name	Data Type	Length	Field Explanation	Primary Key	Default Value
id	bigint		Primary key	Primary key	
addtime	timestamp		Creation time		CURRENT_TIMESTAMP
sheng	varchar	200	Province		
shi	varchar	200	City		
qu	varchar	200	District		
xuexiaoming	varchar	200	School Name		
xingzhi	varchar	200	Nature		
leixing	varchar	200	Type		
zuidifen	varchar	200	The lowest score		
zuidiweici	varchar	200	The lowest rank		
biaoqian	varchar	200	Label		
pici	varchar	200	Batch		
xueke	varchar	200	Subject		
zhuanYe	varchar	200	professional		
xuexiaotupian	longtext	4294967295	School picture		
tianbaoshijian	datetime		Application time		
xueshengxuehao	varchar	200	Student ID number		
xueshengxingming	varchar	200	Student name		
xingbie	varchar	200	Gender		
shouji	varchar	200	mobile phone		
touxiang	longtext	4294967295	Avatar		
crossuserid	bigint		Cross-table user ID		
crossrefid	bigint		Cross-table primary key ID		

4.4 Implementation of the System Module

4.4.1 Visualization module of university information data

The visualization of college information data is achieved by introducing the ECharts library[9]. Firstly, this.\$http is utilized to obtain the total number of colleges, provincial groupings, and other data. Then, the chart container is initialized with echarts.init, and the chart options are constructed by combining preset configuration items (such as titles, legends, and tooltips). After processing the data into a format suitable for chart display, various types of charts, such as pie charts (distribution of schools by province), line charts (trends of the lowest scores of schools), bar charts (statistics of school types), and funnel charts (distribution of disciplines), are rendered through setOption. At the same time, adaptive scaling is achieved through window listening, and some charts also add data scrolling animations, enhancing the intuitiveness and dynamic effects of data presentation, as shown in Figure 12.

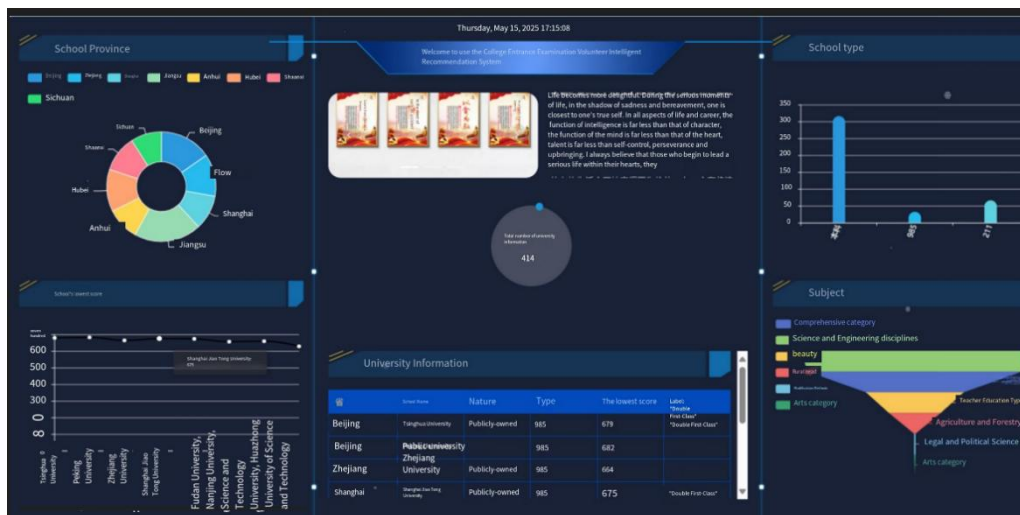


Figure 12 Data Visualization Module for University Information

4.4.2 University recommendation module

Firstly, retrieve the current user's username, denoted as 'userName', from the requested session, and obtain the number of recommended items, denoted as 'limit', from the request parameters.

Secondly, fetch all the college application records, named 'zhiyuantianbaoList', from the database. Then, transform these records into user ratings for majors, represented as 'ratings'. Subsequently, create a 'UserBasedCollaborativeFiltering' object, named 'filter', using the collected user rating data 'ratings'. Invoke its 'recommendItems' method to recommend 'numRecommendations' majors for the target user 'targetUser'. It is advisable to print the recommended major information to the console to facilitate debugging.

Next, construct query conditions using the list of recommended majors, 'recommendations'. Filter out the universities offering these majors from the database and sort them according to the order of the recommended majors. If the number of filtered universities is less than 'limit', select additional universities from the remaining ones in descending order of their IDs to make up the required quantity. Conversely, if the number exceeds 'limit', extract the first 'limit' universities. Finally, encapsulate the processed university list within a 'PageUtils' object and return it to the front - end. The university recommendations for different users are presented in Figure 13.

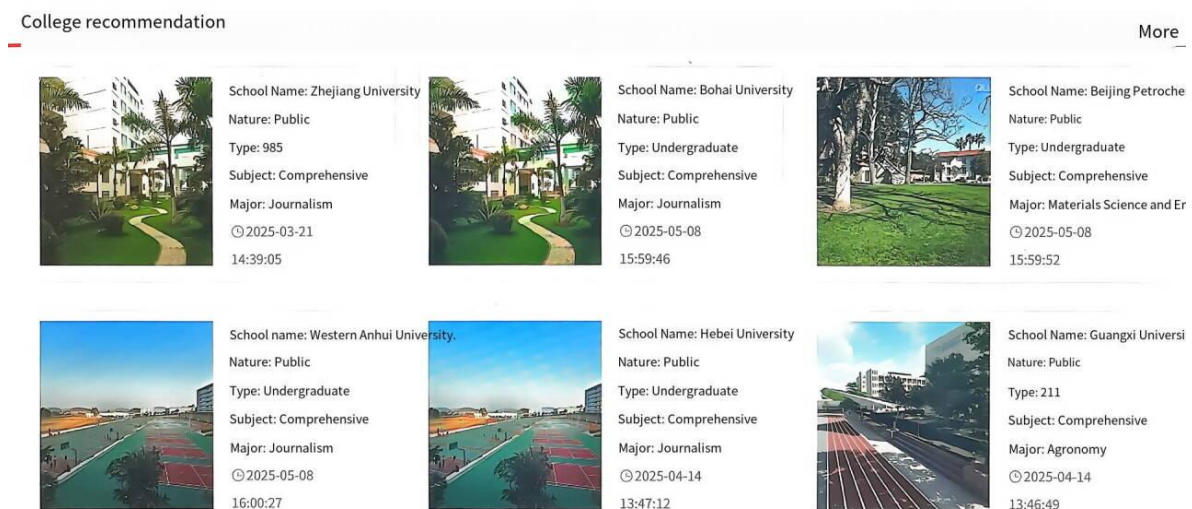


Figure 13 University Recommendation Interface

4.4.3 Module for filling in college application forms

Initially, the front - end page initiates an HTTP request to invoke the interface of the control layer. For instance, when conducting a list query, it calls the /zhiyuantianbao/page interface, as depicted in Figure 14. The control layer accepts the request parameters and invokes the methods of the service layer to carry out business logic processing. Subsequently, the service layer invokes the methods of the data access layer to execute database operations. The data access layer then interacts with the database by leveraging the methods provided by MyBatis - Plus, and subsequently returns the data to the service layer. Ultimately, the service layer returns the processed result to the control layer. The control layer encapsulates this result into a standardized response format and transmits it back to the front - end.

Figure 14 College Application Form Filling Page

4.4.4 AI intelligent Q&A module

When users input questions in the front - end interface and click the send button, the sendMessage method is triggered either when the send button is clicked or the Enter key is pressed. Initially, this method checks whether the user input is empty and whether a message is currently being sent. If these conditions are met, the user's message is added to the messages array. Subsequently, a GET request is sent via axios to the /ai/generate interface of the back - end system. Depending on the request outcome, the AI's response or an error message is appended to the messages array. Finally, the input field is cleared, the sending status is updated, the current conversation is archived in the history record, and the chat window is scrolled to the bottom.

When the front - end sends a question to the /ai/generate interface of the back - end, the generateStream method processes the GET request to /ai/generateStream, taking the user - input message as a parameter. A Prompt object is instantiated to encapsulate the user's message. Then, the chatModel.stream method is invoked to obtain the large - model's response in a streaming fashion.

Upon receiving the request, the back - end invokes the OpenAiChatModel to interact with the large model and retrieve the response. The generate method manages the GET request to /ai/generate, accepting the user - input message as a parameter. It calls the chatModel.call method to forward the user's message to the large model and encapsulates the large - model's response in a Map for transmission back to the front - end.

After the back - end returns the response to the front - end, the front - end presents the AI's response in the chat window and saves the current conversation in the history record. The loadHistory method is designed to load a specific historical record. It assigns the messages within the historical record to the messages array and scrolls the chat window to the bottom. The startNewChat method is utilized to initiate a new conversation. It clears the current conversation messages and sends a welcome message. The interface of the AI - based intelligent question - answering system is depicted in Figure 15.

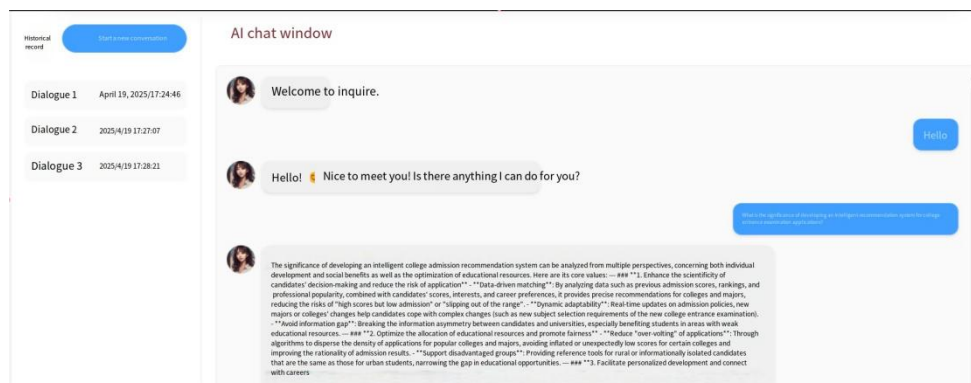


Figure 15 The AI - powered Intelligent Question - Answering Page

5 CONCLUSIONS

In order to address the issues of information asymmetry, over - reliance on experience, and inefficiency during the college entrance examination (CEE) voluntary application filling process, a system has been meticulously designed and implemented. This system is built upon Spring Boot for the backend technology, Vue.js for the frontend technology, and MySQL database. It comprehensively applies the B/S architecture concept and the user - based collaborative filtering algorithm to provide personalized recommendations tailored to users' behaviors and requirements for filling college entrance examination applications[10]. Additionally, it integrates the DeepSeek - V3 large model to construct an intelligent question - answering mechanism, enabling users to pose questions in natural language and facilitating their decision - making process for more optimal choices.

Functionally, the system encompasses several key components: user registration and login, college information query, college recommendation, and simulated application filling. Among these functions, the college recommendation feature stands out as a major highlight. By leveraging advanced techniques such as collaborative filtering algorithms, matrix factorization, and deep learning, and incorporating the relationships between nodes in the knowledge graph[11], this system is capable of generating personalized and accurate college application recommendations that meet the specific requirements of examinees[12]. These recommended colleges exhibit characteristics of "being ahead of others", "stability", and "guarantee", which can significantly enhance the probability of examinees being admitted.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

FUNDING

This work was supported in part by the Guangxi Teaching Quality and Teaching Reform Project "Exploring the Teaching Reform of Computer Programming Courses Empowered by AIGC in the Context of Digital Education" (2024JGA418); Nanning University's 2024 Specialized and Creative Integration Demonstration Course Project "Python Data Analysis" (2024XJYYZ01).

REFERENCES

- [1] Sun Haoran, Wu Xueming, Ji Xueyun. Design and Implementation of an Intelligent Recommendation System for College Entrance Examination Voluntary Filling. *Computer Knowledge and Technology*, 2023, 19(09): 41-45. DOI: 10.14004/j.cnki.ckt.2023.0427.
- [2] Gao Ying, Qi Hong, Liu Jie, et al. Collaborative Filtering Recommendation Algorithm Combining Likelihood Relationship Model and User Levels. *Journal of Computer Research and Development*, 2008(9): 63-69.
- [3] Huang Chuanlin, Lu Yanxia. Research on Hybrid Music Recommendation Algorithm Based on Collaborative Filtering and Tags. *Software Engineering*, 2021, 24(4): 10-14.
- [4] Wang Xuedi, Liu Shijian, Wang Yujie. Application and Research of AI Intelligent Question-Answering System in Archival Consultation Services. *Shaanxi Archives*, 2023(02): 18-20.
- [5] Xavier A, Ananth S, Jie B, et al. Transformer Models: An Introduction and Catalog. 2024. <https://arxiv.org/pdf/2302.07730.pdf>.
- [6] Lyu Meng, Zhang Wei. Design of Hospital Information Management System Based on Cloud Platform and MVC Architecture. *Automation Technology and Application*, 2022, 41(6): 148-151.
- [7] Gao Fei, Luo Qunying, Tian Lei. Web Application System Testing. *Modern Information Technology*, 2019, 3(19): 106-108. DOI: 10.19850/j.cnki.2096-4706.2019.19.034.
- [8] Wan Hongyu. Design and Implementation of College Entrance Examination Voluntary Filling Auxiliary Platform Based on SpringBoot. *Capital University of Economics and Business*, 2022. DOI: 10.27338/d.cnki.gsjmu.2022.000643.
- [9] Li Pan. Design and Implementation of a College Entrance Examination Voluntary Filling Analysis System. *Huazhong University of Science and Technology*, 2019.
- [10] Cao Jinxin, Ju Xiaolin, Chen Xiang. Application of Collaborative Filtering Recommendation Algorithm in the Teaching of Database Principles and Applications. *Computer Education*, 2025(04): 197-201. DOI: 10.16512/j.cnki.jsjy.2025.04.043.
- [11] Qin Z ,Wu D ,Zang Z , et al. Building an intelligent diabetes Q&A system with knowledge graphs and large language models. *Frontiers in Public Health*, 2025, 13 1540946-1540946.
- [12] Yu Jianglong, Song Tengfei, Wang Dechao, et al. Review of Research Methods for Personalized Recommendation Systems. *Computer Knowledge and Technology*, 2024, 20(10): 46-49. DOI: 10.14004/j.cnki.ckt.2024.0483.