Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 11, No. 3, 2025

Design of an LLM-Driven Personalized Learning Resource Recommendation System

-- A Comparative Study

Zhiyuan Ma, Jie Wu

School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan Liaoning, 114051, China

Abstract: Recently, the demand for personalized learning resources has increased substantially. Traditional recommendation algorithms, such as collaborative filtering (CF), often struggle with cold-start scenarios and data sparsity, limiting their applicability in new platforms. In order to address these challenges, this study proposes a personalized learning recommendation system based on large language models (LLMs), which use their advanced understanding capabilities to analyze user intent and generate recommendations. The system architecture employs the Spring Boot and Vue frameworks for efficient frontend-backend development, while the LangChain framework on LLM interactions. The integration of fine-tuned prompts with user-generated tags and historical learning data enables the platform to dynamically generate recommendations through APIs of LLMs, such as ChatGPT and DeepSeek. This approach effectively addresses the issue of suboptimal recommendation accuracy caused by data sparsity within the platform. However, limitations such as hallucination phenomena and API cost scalability necessitate further optimization. It is evident that the platform will require continuous optimisation in the future.

Keywords: Recommendation System; Large Language Model; LangChain Framework.

1. Introduction

The continuous growth of the online education industry has precipitated a surge in demand for high-quality educational resources among university students. In this context, the concept of personalized learning has emerged as a pivotal research domain in educational technology, attracting considerable attention from both academic and industrial communities. It is evident that online education websites such as MOOC and Coursera have recently begun to place a greater emphasis on personalized learning. However, in the face of massive learning resources, plenty of students are difficult to quickly find suitable materials for themselves. Therefore, it's necessary and feasible to build an online study platform which integrates various high-quality resources and is easy to access and utilize.

Although the research of personalized learning is deepening, many learning platforms still face the problem of insufficient accuracy and adaptability of recommendation systems in practical applications.

This study posits that a systematically optimized recommendation system should be incorporated as an essential component of contemporary learning platforms. Motivated by this proposition, the investigation explores the technical feasibility of constructing a personalized learning recommendation platform with large language models (LLMs).

Building on this foundation, this study aims to develop a personalized learning recommendation platform based on the Spring Boot and Vue frameworks, with a dual focus on enhancing recommendation accuracy and user experience through analysis of LLM-based recommendation algorithms and conventional collaborative filtering approaches.

In terms of methodological approach, this learning platform employs a Transformer-based large language model architecture to construct a dynamically recommendation system. Experimental demonstrate that the LLM-based recommendation algorithm significantly outperforms traditional recommendation algorithms in certain domains, such as cold-start scenarios.

2. System Design

2.1. System Requirement Analysis

This paper focuses on college students as the research subjects, constructs a knowledge database based on relevant knowledge points of computer science students, and designs and implements a personalized learning recommendation platform based on large models. From the perspective of college students, the majority are confronted with a lack of clarity in selecting suitable courses for themselves.

To solve this problem, this recommendation system needs to determine user profiles with their feedback and push learning resources they would be interested in. Furthermore, when confronted with voluminous learning resources, the system necessitates structured annotation and hierarchical categorization. During the initial system deployment phase, manual annotation can be employed to establish primary classification schemata. Upon reaching critical data thresholds, these structured knowledge units should be organized into table formats for processing through large language models.

Therefore, the system architecture should be divided into student users and administrator users. For learner-end implementation, user interfaces should be provisioned with some CRUD (Create, Read, Update, Delete) operations for course management, complemented by other learning features including course evaluation, bookmarking, and commenting functionalities. Administrators are able to operate the majority of objects in the system, including those pertaining to student management, course information management, course comment management, administrator user management, course order management, and personalised learning analysis.

To better achieve the aforementioned operations, the system architecture implements a B/S(Browser-Server) architecture. And the backend infrastructure uses Spring Boot framework with Maven-based dependency management for modular component integration and version control. Frontend implementation combines fundamental web technologies (HTML5, CSS3, ECMAScript 2020) with Vue3 and Element-UI component library for rapid interface prototyping. The persistence layer employs MySQL relational database configured with Navicat Premium's visual administration toolkit (PremiumSoft, 2022) for schema optimization. The development environment is based on IntelliJ IDEA, selected for its native support of Spring Boot application profiling, MySQL connectivity via Database Navigator plugin, and Tomcat server integration. Lastly, in the recommendation system, Python is used to access large LLM and perform data handling operations.

2.2. Functional Requirement Analysis

The architecture of the system is fundamentally divided into two primary modules: the front-end student user module and the back-end administrator module, as illustrated in Figure 1 below.

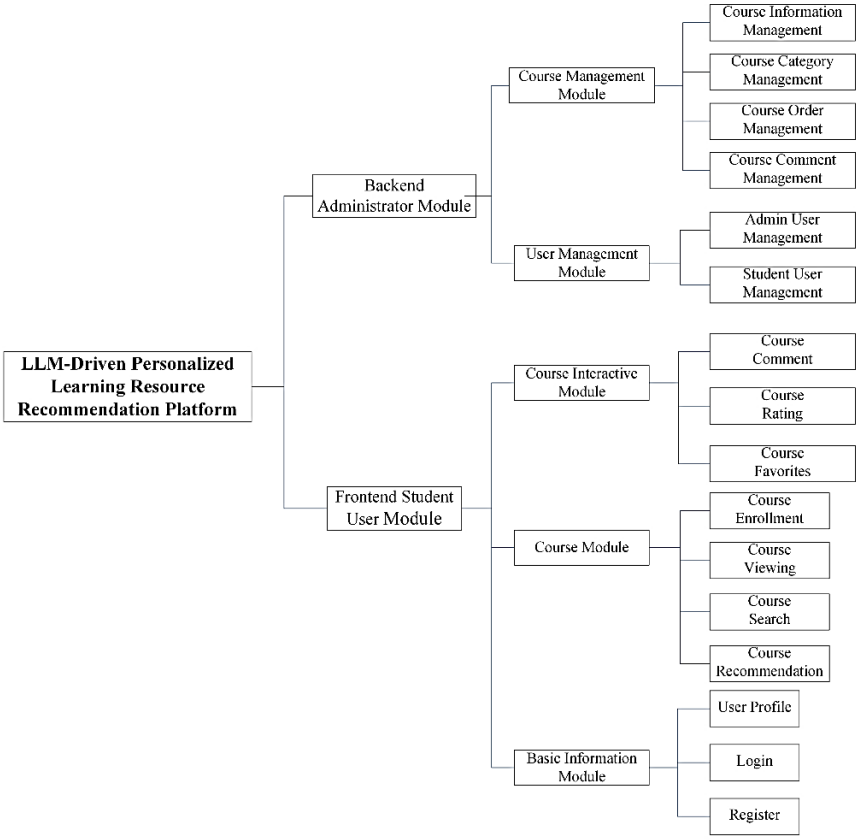

Figure 1. The primary functional modules of the system

2.2.1. Frontend Student User Module

User login and registration allows users to register using their mobile phone number or email address. To prevent a batch of registrations by bots, a captcha is added, and account passwords are stored using Base64 encoding.

The Personal Course Management function is divided into several functions: search, purchase, study, and course drop. The course purchase module offers comprehensive course details, including the introduction, duration, and syllabus, to guide users' purchasing decisions. The course search function supports both fuzzy and exact searches, enabling users to find desired courses via keywords. The course study module includes watching courses, writing reviews, and adding courses to favorites. When a user requests to drop a course, the system properly handles the order, clearly informing the user of refund rules and timelines.

2.2.2. Personalized Recommendation System

The system analyzes user's historical learning data and the tags selected by the user during registration, and combines it with a big model to generate a personalized learning resource recommendation list. Users can feedback their needs to the big model through the chat module, LLM receives the user information and generates a list of recommended learning resources to the system, which receives the list for visualization and sends it to the user.

2.2.3. Backend Administrator Module

The administrator mandates critical system operations through privileged account permissions, including user management, course information management and course order management.

3. Design of the Recommendation Module

3.1. An analysis of the System's Related Technologies

3.1.1. Construction with LangChain Framework

LangChain, a framework for developing applications powered by large language model (LLM)[1], is favored by lots of developers for its rich functionality, highly modular design, and extensive community support. It offers a suite of components, interfaces, and tools that allow for the rapid

development of LLM - based applications. Therefore, LangChain was chosen as the framework for building this recommendation system. Leveraging its components, we can quickly integrate the designed prompts with other relevant recommendation - system information to enable LLM - generated recommendations.

3.1.2. Processing User Data with Embedding

In natural language processing (NLP), “Embedding” is commonly used to transform textual data into vector form, which can approximately reflect the semantics of the text. Consequently, this system selected Text2Vec [1], an open - source project, to perform text similarity calculations. Through its built - in models, such as Word2Vec and BERT, it can efficiently evaluate the semantics and similarity of texts.

3.1.3. Leveraging Existing Large LLMs for Processing

With the continuous iteration of LLM in the past few years, the mature LLMs on the market, such as ChatGPT-4o, DeepSeek-R1, Claude3.5-sonnet, are fully capable of text processing for some professional applications. Therefore, this system accesses ChatGPT.4o-mini and deploys the API of DeepSeek-R1-671B to realize efficient and accurate text processing.

3.2. How to Integrate Large Language Models

3.2.1. Prompt Tuning

In recommendation systems and LLM applications, "Fine - Tuning" and "Prompt Tuning" is essential to enhance the LLM's task performing ability.[2] Since the platform is implemented by connecting to existing LLMs through an API for recommendations, "Prompt Tuning" is used to fine-tune the LLM. In this platform, here's a general process:

Firstly, manually collecting or creating several task examples related to the recommended content. Testers optionally include recommendation logic or results. And Send these examples to the LLM for guidance. After the LLM samples its dataset and outputs results, observe the outputs for the next Prompt adjustment.

Secondly, the system designs an appropriate Prompt and inputs user information and context as auxiliary information to clarify the specific task of the Prompt and the specific recommendation target. For example, on a learning platform, the recommendation target should focus on courses or learning resources. After clarifying the task, the system defines it as a sequence to sequence (Seq2Seq) problem and start writing the Prompt. When writing the Prompt, the system use different delimiters to distinguish user information, instruction information, and context information, and add sample examples to show the correct output format to the LLM. When writing the Prompt, different separators are used to differentiate the user information, instruction information and context information, and sample examples are added to show the correct output format to the LLM.

Lastly, the LLM returns the output results to the system in a standardized format such as JSON, allowing the system to deliver the results to the user in a specific format.

3.2.2. Recommendation System Implementation

In the implementation, users can provide tags, typically selected from a sufficient number provided by the system, to indicate their desired learning content. The system processes these tags for Prompt generation and sends them to the LLM. The LLM then generates a user profile based on the Prompt, and outputs learning resources from the existing dataset that may interest the current user, fulfilling their needs.

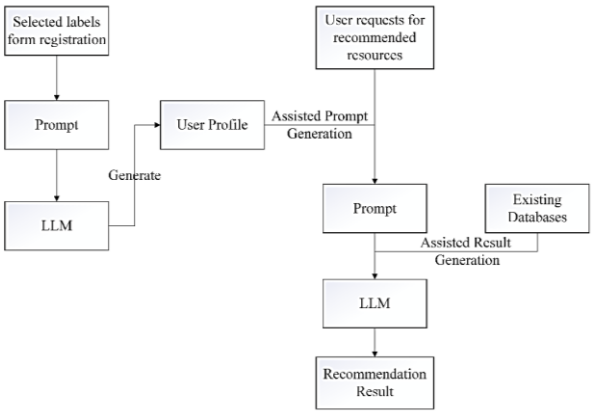

Figure 2. Flowchart of the Recommendation System

4. Comparation with Traditional Recommendation Algorithms

Compared to traditional recommendation algorithms, recommendation algorithms based on LLMs have unique advantages and limitations.

Traditional recommendation algorithms, such as Item-based Collaborative Filtering (ItemCF) and Matrix Factorization (MF), generate recommendations by analyzing user behavior and item feature data based on a certain amount of data. These algorithms perform well when dealing with large amounts of data with clear user or item features, producing satisfactory recommendations. However, their accuracy suffers from significant degradation in data-scarce scenarios, particularly during cold-start situations. In contrast, LLMs pretrained on massive corpora exhibit superior cold-start capabilities through their inherent comprehension of cross-domain knowledge and contextual semantic relationships. For instance, in this learning platform implementations, LLM-enhanced recommendation systems can leverage minimal user-provided tags collected during initial onboarding processes to suggest personalized learning resources. This capability substantially mitigates data sparsity challenges faced by emerging platforms.

In addition, large language models, which require no local computational resources, can be deployed on virtually any platform, ensuring a broad deployment scope. Developers need not consider local computational resources for training and deploying recommendation models, which reduces the usage threshold of recommendation platforms. Moreover, LLMs can analyze user needs through the specific language they input. Users can describe their needs in natural language, which large models parse to more accurately grasp user intent. This enables the provision of more personalized recommendation results, enhancing the user experience.

However, LLMs are still affected by the "hallucination phenomenon". This significant drawback directly causes LLMs to sometimes generate incorrect recommendations, providing users with misleading information.[2] For example, LLMs may generate learning resources that do not exist in the database or recommend resources that appear relevant to user needs but actually do not meet their requirements.

In contrast, existing mature LLMs require API invocation, which incurs cost overheads, especially when user interaction volume is large, making the invocation cost substantial. This is a significant concern for small and medium-sized platforms. In contrast, traditional recommendation algorithms such as Item-CF and MF, while requiring substantial computational

resources and extensive training datasets during the model initialization phase, exhibit negligible marginal costs in subsequent recommendation operations. This characteristic emerges particularly during the inference stage, where these memory-based methods primarily rely on precomputed embeddings or similarity matrices rather than requiring real-time neural network computations.

Another aspect to consider is the uncontrollability of the outputs from large models. When aligning LLMs with recommendation objectives through parameter-efficient fine-tuning, practitioners must accept a partial compromise of their general-purpose capabilities.

5. Conclusion

The personalized recommendation system integrated into this learning platform leverages LLMs to achieve accurate alignment of educational resources.

Experiment evaluations have demonstrated that LLMs exhibit notable accuracy and reliability in generating recommendations, proving competent for resource recommendation tasks in most operational scenarios.

However, while current LLM-based recommendation systems demonstrate superior semantic comprehension and cold-start capabilities, they still confront critical technical challenges requiring resolution. The most significant issue is the inherent hallucination problem in LLMs, which directly impacts the accuracy and the reliability of recommendations. Conventional strategies typically implement multi-constraint injection through Prompt-Tuning, but this approach notably increases request token consumption, leading to huge API cost escalation. Future iterations of this platform will systematic optimizations on these technical limitations.

Acknowledgments

This work it was supported by National College Students' innovation and entrepreneurship training program project.

References

[1] Zhang Xiyun, Tan Kun, Ouyang Taohui, et al. Design and implementation of a personalized exercise recommendation system based on large language models[J]. Digital Technology and Application,42(07), 32-34.

[2] Wang Minghao, Yin Tao, Yang Hongjie, et al. Development and application of knowledge graph and large model technologies [J]. Cyber Security and Data Governance 2023, 42 (S1):126-131.

[3] Wu Guodong, Qin Hui, Hu Quanxing, et al. Research on large language model and its personalized recommendation[J].CAAI Transactions on Intelligent Systems,2024,19(06):1351-1365.