Course Recommender

Final Report

Contents

1 Introduction

Course selection is a critical decision-making process for students, influenced by factors like interests, workload preferences and compatibility with instructors. Existing systems often lack personalization and contextual relevance.

NeuroLearn is an AI-driven Course Recommendation System designed to address these limitations by delivering personalized suggestions aligned with individual student needs. The system leverages AI models to understand student queries and retrieve relevant course information efficiently.

Objective: To develop an intelligent system that provides tailored course recommendations using historical performance, student interests, peer reviews, and professor compatibility. It supports workload balancing and real-time filtering to enhance usability and decision-making.

2 Problem Statement

Current course recommendation systems lack the ability to provide personalized suggestions that consider a student's academic profile, workload preferences, and instructor compatibility. We seek to develop a sophisticated Course Recommendation System that provides students with personalized course suggestions tailored to their academic goals, workload preferences, and interpersonal dynamics with professors. By leveraging cutting-edge AI/ML technologies, the system aims to enhance the student experience, improve academic outcomes, and foster meaningful professor-student relationships.

3 Novelty & Differentiation

3.1 AI-Powered Personalization

- Our system uses **BERT & Qwen2.5-14B-Instruct** to interpret student queries and contextually retrieve recommendations.
- Considers multiple dynamic factors (*workload, professor fit, peer reviews*) instead of just prerequisites or ratings.
- First system to let students explicitly define workload preferences (*"Easy," "Medium," "Hard"*) and balance courses accordingly.
- Uses **NLP** on student reviews to assess teaching style (*lecture-heavy, project-based, flexible grading*) and matches it with individual preferences.

3.2 Comparison to Existing Solutions

- **Traditional Systems** (e.g., University Catalogs): Static, filter-based, no personalization.
- **Peer Review Platforms** (e.g., RateMyProfessors): Limited to ratings, no workload or AI-based matching.
- **Basic Recommenders** (e.g., Degree Planners): Rule-based, lack contextual understanding.

4 Use Case Selection

4.1 Primary Users

Students seeking optimal course selections aligned with their academic goals, workload preferences, and instructor compatibility. Academic advisors who assist students in planning their semesters more effectively.

4.2 Key Scenarios

1. **Balanced Workload Planning** A student wants a manageable semester and selects ”Medium Workload” preference. The system recommends courses that fit their desired difficulty level while avoiding scheduling conflicts.
2. **Professor Compatibility Matching** A student prefers interactive, discussion-based teaching styles. The system analyzes professor reviews and suggests instructors whose methods align with the student's learning preferences.
3. **Interest-Based Course Discovery** A computer science student wants to explore AI-related electives. The system recommends relevant courses based on their past coursework, grades, and peer reviews.

5 System Architecture and Workflow

Neurolearn's architecture uses the modular way to implement the ideas to prevent unnecessary memory wastage, saves time and present the best recommendationto the user. It consists of three main components: the frontend (client), the backend (API server), and the database.

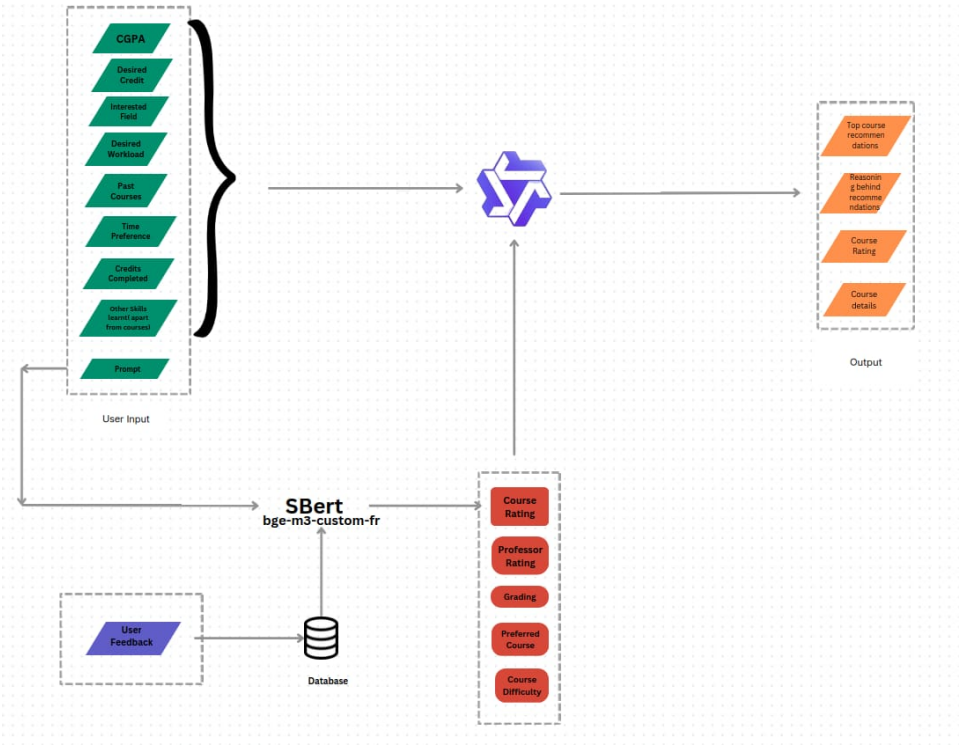

Figure 1: NeuroLearn's System Architecture

1. Database (PostgreSQL)

A PostgreSQL database forms the backbone of the system storing all the necessary data required for the analysis of the relationships between data and use it provide the required service for the user. Our database holds User profiles and login details and Course metadata (titles, descriptions, categories).

PostgreSQL was selected because it handles structured and semi-structured data really well, which is useful for storing course metadata, user preferences, and embeddings. It's reliable and secure, making it ideal for managing user accounts and history.

2. Bert Model (Fine-tuned BGE-M3)

The system uses a fine-tuned BGE-M3 model to generate high-quality sentence embeddings for both user queries and course descriptions. This allows the system to understand the semantic meaning behind student prompts, even when they're expressed in diffeent ways to provide the best course for the user.

We selected BGE-M3 because it is ranked highly on the Massive Text Embedding Benchmark and is optimized for multilingual retrieval tasks. It is fast, accurate, and can effectively capture the intent behind complex course selection queries.

3. Language Model (Qwen2.5-14B-Instruct)

To provide reasoning and natural language explanations for recommendations, we integrated Qwen2.5-14B-Instruct, a powerful large language model known for its instruction-following capabilities.

Qwen is used to interpret the output of the embedding model along with user preferences and course metadata. It produces human-like justifications and summaries to help users understand why a course was recommended. Our thought process behind this was to improve user trust and makes the process more transparent.

4. Frontend

The frontend is developed with React.js, giving the user interface a dynamic and responsive touch. It enables the users to sign up and login successfully using authentication.

It provides flawless interaction with the user in such a manner that components are in modular form so it can be reused for further projects. It provides a straightforward UI that allows the user to access the main feature and use it accordingly.

5. Backend (API Server)

The backend is implemented using FastAPI, a high-performance Python web framework. It is used for user registration and authentication, for the submission of course recommendation prompts by the user and the retrieval of the recommended course. The backend demonstrates proper interaction between Bert Model, Large-Language Model, and the database.

Workflow Overview

User Input: The user types in a free-text query stating their target, interests, or preferences (Example - ”I want an easy computer vision course”).

Semantic Search through Fine-Tuned BERT: The input is embedded through a finetuned BGE-M3 model selected after going through its performance and ranking on Massive Text Embedding Benchmark Leaderboard. This model represents both the user prompt and all course descriptions as dense vectors. Cosine similarity is calculated between the prompt and each course vector to determine the best course according to the prompt.

Best Match Selection and Metadata Retrieval: A single course with the highest similarity score is chosen and its metadata (like difficulty level) is fetched from the PostgreSQL database.

Natural Language Reasoning through Qwen2.5-14B-Instruct: The user input and chosen course are input to the Qwen2.5-14B-Instruct model. The large language model (LLM) produces a reasoning explaining why this course fits the user’s preferences.

Frontend Rendering: The backend responds with a answer that includes the selected course and the synthesized explanation. The frontend shows this data in an easy-to-understand structure, giving users its recommended course.

6 Model Selection and Justification

BERT Model - For **semantic similarity** and **course matching**, we utilized the **BGE-M3** model from the **SentenceTransformers** library. The model was a top performer in the **Massive Text Embedding Benchmark (MTEB)** for tasks such as text retrieval and semantic textual similarity (STS), which suited

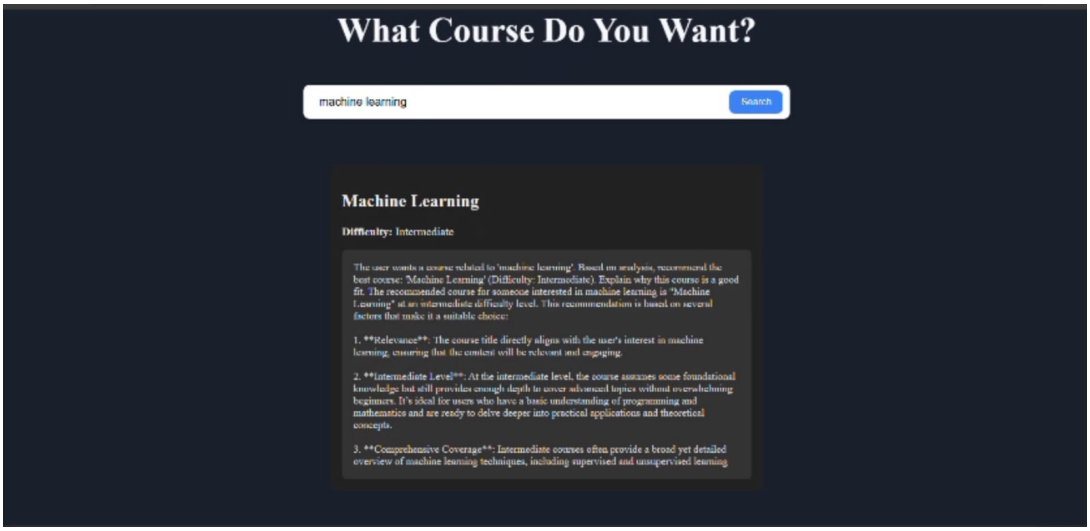

Figure 2: Frontend Query with Response

our requirement well. Its high performance on multilingual datasets and general-purpose embeddings positioned it as a strong choice for understanding user queries and course descriptions.

LLM Model - To process natural language reasoning and produce explanations, we employed the **Qwen2.5-14B-Instruct** model in its **8-bit quantized** version. This variant balances performance with resource efficiency, enabling deployment on lower-end hardware while maintaining high-quality output.

7 Future Improvements

Although NeuroLearn provides a robust AI-driven framework for personalized course recommendations, several enhancements can further improve its effectiveness and user experience.

- **Introducing Course Reviews and Ratings:** Integrating detailed student reviews and quantitative ratings can enrich the recommendation process by providing real-time feedback on course quality, instructor effectiveness, and workload accuracy. Sentiment analysis in reviews could help dynamically adjust course difficulty and professor compatibility scores.
- **Enhanced User Profiling:** Expanding user profiles to include detailed academic history, learning styles, and career goals would allow more granular personalization, adapting recommendations to evolving student needs.
- **Real-Time Enrollment and Availability Tracking:** Adding live updates on course seat availability and scheduling conflicts would enable dynamic recommendations that adapt instantly to changes, improving planning efficiency.

8 Conclusion

While building NeuroLearn, we explored how AI can help make course planning more personal by matching student interests with the right courses. Using BGE-M3 for understanding course and query meaning, and Qwen2.5-14B-Instruct for reasoning, our system could understand what students were

really looking for, not just keywords, but things like how hard a course is, the professor's teaching style, and what the student wants to learn.

This project showed that choosing courses isn't just about searching a database. It's a personal decision. By combining language understanding with smart logic, we created a system that gives more meaningful and relevant suggestions. This lays the groundwork for even better tools to support smarter, more thoughtful academic planning.