How Can We Use LLMs for EDM Tasks? The Case of Course Recommendation

Md Akib Zabed Khan[1,*], Agoritsa Polyzou[1,*] and Neila Bennamane[1]

[1]Florida International University, Miami, Florida, USA

Abstract

Choosing appropriate courses for a semester is a challenging task for undergraduate students. To facilitate the course selection process, different course recommendation systems have been proposed implementing different machine learning algorithms and techniques. At the same time, recently, there has been a rapid development of Large Language Models (LLMs) (e.g., GPT4, Llama3, and Gemini), which have been adopted in numerous applications and have influenced all walks of life. In this paper, we explore their potential to assist stakeholders in higher education with course recommendation. We explore two different ways to directly (and offline, to ensure no sensitive data leakage) generate recommendations from pre-trained or fine-tuned LLM models. We also propose a novel ChatGPT-assisted course recommendation model (GPTaCR) which follows a different methodology. It utilizes the output of ChatGPT to form rules that capture relationships among courses. Based on these rules, we generate a set of courses for every student to take next, given their prior enrolment history. We use a real-world dataset to evaluate the performance of our proposed models compared to other relevant course recommendation approaches. The findings highlight that our proposed manages to best combine the rich knowledge base of ChatGPT with information about past students' enrollment history. We hope that this work can be a source of motivation for researchers to look into how LLMs might improve recommendation performance and other educational data mining tasks.

Keywords

LLMs, Course Recommendation, Course Descriptions, Undergraduate Education,

1. Introduction

In higher education, one of the difficulties faced by students is to select a set of courses every semester and balance the course workload within a semester. While selecting courses, they need to consider degree requirements, prerequisites of courses, their interests and career choices, and which courses will build their base knowledge to take more advanced courses in the future. So, choosing a good set of courses is a non-trivial task for students. They may get assistance from the departmental course catalog or other senior students. However, their insights might not be personalized to the preferences, experience, and background of a particular student. Academic advisors can also help, but in most institutions, the high ratio of students to advisors limits the time and dedication an advisor can devote to a single student, consequently limiting the usefulness of their interactions [1]. Due to the lack of proper guidance, students may not select appropriate courses, which might have adverse effects on their time-to-degree or retention.

The development of data mining and machine learning models can assist student advising by generating personalized course recommendations for each student based on insightful analysis of historical data records of past students. For example, researchers propose different course recommendation systems building Markov chains [2] or deep learning models [3, 4] to analyze students' course enrollment data and student-course interactions. Besides, natural language processing (NLP) approaches analyze the topics and vocabulary available in course descriptions to understand the relationship of different courses [5, 6] and capture the students' preferences by inspecting student-course interactions [7].

The fast development of large language models (LLMs) and their chatbot prototypes like ChatGPT3.5 and ChatGPT4 has increased the potential to use NLP approaches for solving different real-world problems. After the release of ChatGPT3.5 in November 2022, generative artificial intelligence (GenAI) has been widely used in different aspects of life. For example, according to recent articles, 45% job seekers using GenAI to polish, update, and improve their resumes [8], students feel more comfortable asking an LLM for explanations rather than talking to a professor or teaching assistant [9], companies improve their customers' experiences by providing personalized feedback [10], etc. For the task of recommendation, different prompting strategies in LLMs have been investigated to recommend relevant products [11, 12]. The use of ChatGPT has been proposed for tasks within the educational data mining (EDM) domain, as well. ChatGPT has been explored for generating mathematics assessment questions [13], programming learning [14], fixing programming bugs [15], and major recommendation to students [16]. To the best of our knowledge, no one has explored the potential of utilizing ChatGPT for course recommendation tasks.

In particular, we explore different ways to use LLMs for the task of course recommendation. We start by using simple, offline, pre-trained models directly for course recommendation. Next, we consider fine-tuning strategies for simpler LLMs to enrich the knowledge base with student enrolment data. Next, inspired by the association rule mining technique introduced in [17, 18] for course recommendation, we explore a rule-based approach where LLMs assist us in forming rules which will then be used for recommendation. In our proposed **ChatGPT-assisted course recommendation** (GPTaCR) system, we generate sets of courses that are frequently taken together by students. By providing these sets of courses and different contexts (course names, acronyms, descriptions), we ask ChatGPT4 to offer us a set of courses well-suited for students to take next. In this way, we form rules and use them to generate recommendations for the students, based on their prior course registration history. We use real-world data from a US public university. Our experimental results indicate that ChatGPT understands the association and relationship of courses by analyzing the provided prompts. It manages to offer meaningful recommendations with minimal context provided, so it can be a

*Corresponding author.

mkhan149@fiu.edu (M. A. Z. Khan); apolyzou@fiu.edu (A. Polyzou); Nbenn017@fiu.edu (N. Bennamane)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-3840/L3MNGET24_paper12.pdf

useful and practical tool to recommend courses efficiently and accurately.

2. Related Work

2.1. Course Recommendation

Researchers propose different course recommendation systems (CRS) for university students by collecting historical course enrollment data from warehouses [17, 19], course descriptions from departments and collecting students' interests, learning goals, and skills by conducting surveys [20, 21]. Numerous CRS have also been proposed to recommend online courses to the users of online course platforms like Moodle, edX, and Coursera, etc. [22, 23, 24].

For course recommendation in university environments using enrollment data, an association rule mining algorithm (Apriori) has been proposed to capture the association of all courses taken by each student [17, 18]. The idea is to treat courses as items and generate confident rules. When a student's prior courses match the left-hand side of a rule, the rule is activated, and the consequent courses are used for recommendation. Matrix factorization models are introduced to recommend courses considering similar students' preferences for one course over another one [25]. Markov chain and deep learning models (i.e., long short-term memory networks) have been used to capture the sequential transitions of courses taken semester-by-semester [3, 4, 2].

Whereas, NLP approaches focus more on textual information (e.g., course descriptions) to calculate similarities between courses by analyzing the topics and knowledge components each course covers. Term Frequency–Inverse Document Frequency (TF–IDF) is the most common technique to analyze course descriptions and find similar courses by analyzing how many times a word appears in the document of each course [5, 6]. On the other hand, Shao et al. propose an NLP-based PlanBERT model to recommend courses for multiple consecutive semesters by considering each student's course history as a sentence where each course is a word [7]. Since NLP-based tools are more advanced after the introduction of LLMs and different applications of conversational modeling, we explore the use of ChatGPT with different contexts (course names, acronyms, descriptions) to provide personalized course recommendations to undergraduate students.

2.2. LLMs in Education

Language models have been applied and explored in education settings for different topics such as providing personalized feedback and assistance to students [26, 27, 16], generating course concepts [28], automatic hint generation [29], automated assessment process [30], grading open-ended questions [31], generating feedback for programming errors [32], detecting student talk moves [33], training teachers [34], nursing training [35], and sentiment analysis task [36], etc. While many applications provide positive insights and findings, there are some limitations of ChatGPT for doing each specific task and outcomes are not as good as the results when humans complete the tasks like feedback generation and assessments. For example, Botelho et al. observe that the range of variations that teachers see while giving students feedback is not captured by ChatGPT by encoding students' responses for comparison [30]. Markel et al. find that there are limitations in the realism of scenarios when teaching assistants get training from the chat system [34]. Lekan et al. cannot provide major recommendations to transfer students using ChatGPT [16]. Moreover, notable privacy, equity, safety, and ethical concerns surface when LLMs are used in educational contexts [37].

instruction: "Recommend next course(s) by analyzing course information and prior courses of each student".

input: "<|user|>: List of course codes with names: ["c_1: Algorithms",...]. Prior course list sorted in semester by semester of student 1: [[c_1,c_2], [c_3,c_4,c_5]]". Recommend next k new course(s) for this student.

Output format: a list of course codes only.

Figure 1: Example of the instruction and input format to get recommendations from the pre-trained Llama-3 model.

3. Proposed Approaches

3.1. Direct Use of LLMs - CRwLLM

First, we use an open-source pre-trained base LLM model to generate recommendations. In this approach, we use the Llama-3-8B-Instruct model[1] provided by hugging face[2] to get inference for our data. Here, we do not use any historical course enrollment data for re-training or fine-tuning the model. We just provide all the course codes with course names and prior course registration history of a student as one instance (input) and ask the Llama3 model (offline prompting running codes) to recommend a set of courses (output) by analyzing the provided information. An example of the instruction and input format for a target student is depicted in Figure 1.

We cannot generate recommendations from the open-source GPT2 [38] model because the input describing an instance (course codes, course names, student's prior courses) exceeds the maximum sequence length for this model.

3.2. Fine-tuning LLMs - CRwFine

Alternatively, we can fine-tune LLMs to enrich their knowledge with historical data relevant to our specific task at hand. A similar approach has also been proposed for item recommendation [12]. We fine-tune two open-source models, Llama-3-8B [39] and GPT2 [38] denoting them as CRwFine(Llama) and CRwFine(GPT), respectively. We opted for these LLMs since we can download and easily use them for free. Alternatively, we could use the APIs offered for fine-tuning, but that involved uploading training data, which we did not want to do. By performing the fine-tuning process completely locally, we do not risk the privacy of student data. Supervised fine-tuning of an LLM includes the following steps.

Data preparation: Following the prior work for item recommendation [12], we use the course registration history of the students to fine-tune an LLM so that it captures the historical enrollment patterns. An example of a training instance we preprocess is illustrated in Figure 2. We use <|user|> and <|assistant|> tokens to indicate input and output in each training instance[3] where input is the prior course list of a student and output is the courses taken in the target

[1]https://ai.meta.com/blog/meta-llama-3/
[2]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[3]https://huggingface.co/blog/llama3

instruction: "Recommend next k new course(s) by analyzing historical enrolment patterns, course information and prior courses of each student".
input: "<|user|>: Prior course list sorted in semester by semester of student 1: [[c_1,c_2], [c_3,c_4, c_5]]".
output: "<|assistant|>: Recommended courses: [c_1,c_2,...,c_k]".

Figure 2: Example of a training instance for fine-tuning an LLM.

(last) semester. We also provide course descriptions as input to fine-tune the model to capture the semantic similarity of courses as described in [40] where there is no output (empty string) for those texts.

Environmental set-up: We use the transformer library of hugging face[4] to access the base pre-trained language models (Llama-3-8B and GPT2-1.5B). We use Autotokenizer to tokenize each word to fine-tune the Llama3 model and GPT2tokenizer for the GPT2 model. We use BitsandBytes and parameter-efficient fine-tuning (PEFT) libraries for fine-tuning efficiently by reducing computing requirements. PEFT library provides LoraConfig to use QLoRa [41] which stands for Quantization and Low-Rank adapters. Quantization shrinks the size of a base LLM model by saving 8 or 4 bits per parameter (we save 4 bits per parameter). Using the QLoRa method, we freeze the existing base LLM model and add some parameter weights (low-rank adapters) to the model to train which requires less memory and GPU support.

Train model and inference We define SFTTrainer with training arguments that include which layers of the base model to apply the adapters (we choose the attention layer because we want to generate attention scores for each course to recommend and update the parameters accordingly) and different hyperparameters like number of epochs, learning rate, etc. Then we train the base model with our dataset and use the fine-tuned LLM model for inference. From validation and test sets, we provide a target student's prior courses as input, keep the output empty, and then generate recommended course list as output from the fine-tuned model.

3.3. Indirect Rule-Based Approach - GPTaCR

The next step we could explore is to use a more recent and powerful LLM, e.g., ChatGPT4. However, we cannot enter students' registration history in ChatGPT4, as that would violate student privacy, so we introduce two alternative approaches. Inspired by the association rule mining (ARM) approaches for course recommendation, we want to generate rules in the form of $X \rightarrow Y$, where X represents a prior course or frequent sets of courses (course sets) and Y represents the output collected by ChatGPT4 when asked to suggest future courses for a student that has taken the courses in X. Based on the overlap of X with the past courses that a student has taken, we generate the recommendations in a similar manner as in ARM. Our GPTaCR model and its steps are illustrated in Figure 3.

In the **GPTaCR-1C** approach, we provide one course as a prior course of a student and ask ChatGPT4 to suggest a set of courses that are relevant to this course by analyzing provided contexts (course names, acronyms, and descriptions). In this way, we generate suggestions for all the available

[4]https://huggingface.co/docs/transformers/en/index

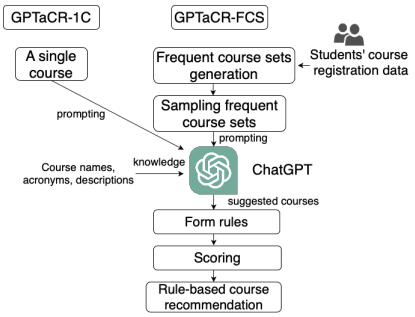

Figure 3: Proposed GPTaCR-1C and GPTaCR-FCS models

courses which means we get 618 rules for 618 courses in our dataset.

For the **GPTaCR-FCS** model, we use the students' course registration history to generate frequent course sets. We consider courses to be the items, and each student's course history over all the semesters is one transaction. Based on all transactions in the training data, we use the concept of ARM to generate frequent itemsets (FI), i.e., the course sets with support greater or equal to a minimum support threshold. Here, each FI consists of one or more courses frequently taken together. We also generate and tested the maximal frequent itemsets (MFI) where all the subsets of a frequent itemset are removed (i.e., if {c1, c2, c3} is a MFI, we do not consider its subsets, i.e., {c1, c2}, {c2, c3}).

While the number of FI (or even of MFI) that we get from the previous step can be quite high, we need to reduce them and make it more practical for us to test those course sets with ChatGPT. At the same time, it is desirable for the course sets to cover as many unique courses as possible. As a result, we cannot simply use a higher minimum support threshold to get fewer FI, as in this case, the coverage of courses will be very poor. What we do is randomly sample course sets from FI or MFI to use them as input in ChatGPT4 prompts. While sampling, we have almost as many unique courses as in the sampled frequent itemsets and the average length of itemsets remains similar to FI or MFI.

Prompt ChatGPT and form rules: For each possible course set X of the previous step, we use ChatGPT4 [42] to generate suggestions for courses to take next after taking the courses in X. We use a zero-shot prompting strategy which means we provide different contexts (i.e., course codes and names, different acronyms of courses, and course descriptions) and a course set X and ask ChatGPT4 to suggest new and diverse sets of courses (Y) that a student could take without provide any examples from the training data. We ask for an estimate of its confidence ($\in [0, 1]$) for each instance. By collecting ChatGPT's output, Y, we form the rules $X \rightarrow Y$ where prior courses of a student are on the antecedent part and the suggested courses from ChatGPT4 are on the consequent part.

A rule gets activated if its antecedent part is matched (using a % of match threshold) by a student's prior courses. From each activated rule, we examine two possibilities regarding courses that we could recommend: first, consider only the courses in consequent sides Y of the activated rules, or secondly, also consider the unmatched courses in antecedent part X of the activated rules.

Scoring rules: We use three different ways to calculate the score of a course to recommend. (i) We use the confidence values estimated by ChatGPT4 for a rule to compute

the recommendation score for each course of the activated rules. (ii) We count how many times an unmatched course appears on any side of the activated rules and the total count is the recommendation score for that course. We denote this scoring rule as the baseline-counting. (iii) We use the matching ratio and a parameter (β) to indicate from which side of the activated rule an unmatched course is recommended to compute the recommendation score of that course. The formula for calculating the score of a recommended course from any side of the rule is as follows:

$$\text{Score}\ (c) = \begin{cases} \alpha + \text{match ratio} * \beta & \text{if } c \in \text{RHS} \\ \alpha + \text{match ratio} * (1-\beta) & \text{if } c \in \text{LHS} \end{cases} \quad (1)$$

where c is the unmatched course in an activated rule, $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are hyperparameters. α captures a course score that should always be added for courses in activated rules. If β is small, then gives a higher score for unmatched courses in the LHS. We sum up the scores of each course over all the activated rules to get the final recommendation score of each candidate course.

Rule-based course recommendation: After using any of the scoring methods, we have a recommendation score for each course. We finally recommend the courses with the highest recommendation scores. If no rule gets activated for a student, we recommend popular courses which were not been taken by that student before.

In ChatGPT4, we create a customized GPT4 chat, write instructions, and examine two cases when providing different contextual information, as shown in 4. In the first case, CNames, we provide only the available list of courses from our training data (i.e., course codes and their names) as context, and ask for suggestions. In the second case, CNames+Desc, we provide additional context by uploading separate files in the knowledge part: the available list of courses, definitions of different acronyms of courses, and available course descriptions. In both methods, we provide 15 students' information at a time, generate suggestions for them and collect the rules. Under additional chat settings in ChatGPT4, there is an option to use conversational data to improve the model which we unselect, as recommendations for later students may get biased and ambiguous by previous course sets and suggested courses from ChatGPT4.

Course names	Course names, acronyms, and descriptions
Assume that you are a course recommender system to support students' course selection in the Computer Science Department. Your task is to recommend relevant courses to students based on courses they took in the prior semesters. Every course has a code, which is composed of three letters that denote the course subject, followed by four numbers. Optionally, there might be the letter "L" at the end of the course code to signify that this code corresponds to a lab. The first digit represents the level for a course, starting from 1 to 5. The list of available valid courses is provided by the file "CourseList.txt" with the following format <course code>: <course name>. Students will enter their information in the following format <"iid": 1, "itemsets":[prior courses]>, which means that the student with id=1 has taken <anonymous> courses in the past. For each student, your task is to recommend <5> new and diverse relevant courses to each student to take in the next semester from the possible <valid course list> by analyzing the course names and estimating your confidence about your recommendation on a scale of [0,1]. Recommended courses for one student should be different from others. Do not repeat your recommendations. Write the recommended course list in 'JSON' record format only with course codes. For example, output format: {"iid": 1, "PC": [prior courses], "RC": [recommended courses], "Confidence": 0.80}. 'JSON' record formatted output should include one line for each record for each student in {}.	Assume that you are a course recommender system to support students' course selection in the Computer Science Department. Your task is to recommend relevant courses to students based on courses they took in the prior semesters and the provided context. Every course has a code, which is composed of three letters that denote the course subject, followed by four numbers. The letter "L" at the end of the course code signifies that this code corresponds to a lab. The first digit represents the level for a course, starting from 1 to 5. The available <valid course list> is provided in the "CourseList.txt" file with the following format: <course code>: <course name>, with one line per course. The definition of some acronyms is provided in the "AcronymList.docx" file in the following format: <acronym>-<explanation>. Different acronyms will be separated by ";". Course descriptions for some courses are provided in the "CourseDescriptions.docx" file in the following format: <course code> <course name> (<course credits>). <course description>, with one paragraph for each course. Students will enter their information in the following format <"iid": 1, "itemsets": [prior courses]>, which means that the student with id=1 has taken <anonymous> courses in the past. For each student, your task is to recommend <5> new and diverse relevant courses for each student to take in the next semester from the possible <valid course list> by analyzing the course names, acronyms, course descriptions, and prior course history of each student and estimating your confidence about your recommendation on a scale of [0,1]. Recommended courses for one student should be different from others. Do not repeat your recommendations. Write the recommended course list in 'JSON' record format only with course codes. For example, output format: {"iid": 1, "PC": [prior courses], "RC": [recommended courses], "Confidence": 0.80}. 'JSON' record formatted output should include one line for each record for each student in {}.

Figure 4: Provided prompts in ChatGTP4.

4. Experimental Evaluation

4.1. Dataset

We collected a real-world dataset from Florida International University, a public university in the US, that includes the course enrollment history of undergraduate students of the Computer Science Department for nearly nine years. As our recommendation should represent good practices, we solely take into account the information of students who have earned a degree, and we eliminate instances where a student had a grade lower than a C in a course. We also eliminate any dropped courses from students' histories. Courses that show up in our dataset fewer than three times are eliminated. Following preprocessing, we have 3328 students' course enrollment histories and 647 distinct courses. Then, we split the dataset into 3 sets: training, validation, and test sets, with 2973, 1231, 657 students and 618, 540, 494 unique courses, respectively. For testing, we use the last three semesters and the preceding three semesters are used for validation and model selection. The remaining semesters are retained in the training set. In the validation and test sets, we eliminate the courses that are not available in the training set because some models can not recommend these new courses. Moreover, we eliminate any students who took courses in less than three semesters, as we require the records from at least the last two semesters in order to produce recommendations for a student. Finally, there are 618 unique courses retained in training, validation, and test sets. Each student may be associated with more than one instance, one for each potential target semester with a prior history length of at least 2 semesters. Additionally, we collect course names, descriptions, and definitions of acronyms for each course in our dataset from the university's undergraduate catalogs.

4.2. Experimental Setup

For fine-tuned LLM models, we follow the experimental setup as described in the Subsection 3.2. The CRwLLM and CRwFine methods did not have any external hyperparameters that we needed to tune.

For the GPTaCR-1C model, we explore different numbers of suggested courses from ChatGPT4 = [3, 4, 5], recommend from only the consequent part (because we have just one course in the antecedent part that is matched with a student's prior course history to activate a rule) and different prior course histories = [all, last 2 semesters, last 1 semester] of each student. We also use two different scoring rules: 1) baseline-counting as described for the GPTaCR-FCS model and 2) calculating the score based on the semester at which the matched course was taken by the student. In this case, we provide higher priority (more weight) if the course was taken in a recent semester.

For the GPTaCR-FCS model, we repeat our experiments three times and report the average statistics and measures in

Table 1
Statistics of frequent itemsets and maximal frequent itemsets for different support values. Here, min supp: minimum support, #c: number of unique courses, and avg len: average length of itemsets.

min	FI			MFI		
supp	#sets	#c	avg len	#sets	#c	avg len
0.05	2080934	71	9.2 ± 2.1	5572	71	9.4 ± 3.0
0.1	128918	48	7.3 ± 1.9	1515	48	7.3 ± 2.4
0.15	23572	43	6.3 ± 1.7	517	43	6.6 ± 2.4
0.2	4212	35	4.7 ± 1.4	276	35	5.5 ± 1.7
0.25	919	28	3.7 ± 1.2	119	28	4.4 ± 1.2
0.3	330	21	3.4 ± 1.3	43	21	3.8 ± 1.6

Table 2
Statistics of average number of frequent itemsets and maximal frequent itemsets after random sampling (minimum support = 0.1). Here, #c: number of unique courses.

#samples	FI		MFI	
	#c	avg length	#c	avg length
700	43.0	7.3 ± 2.0	45.7	7.3 ± 2.4
600	42.3	7.3 ± 2.0	45.3	7.4 ± 2.4
500	42.0	7.3 ± 2.0	45.0	7.3 ± 2.4
400	41.0	7.3 ± 2.0	45.7	7.4 ± 2.4
300	40.3	7.3 ± 2.0	44.7	7.3 ± 2.5

this paper, as we use sampling to limit the number of course sets. We do so to capture any variability of the results caused by the sampling and to ensure that the results are accurate and representative.

For our first step, we apply the Apriori algorithm for frequent itemset generation from mlxtend [43] based on a minimum support threshold. The statistics of FI and MFI for different threshold values are presented in Table 1. We notice that even with the lowest threshold of 0.05 which results, only 71 (out of the 618) courses are present in two million itemsets. Moving forward, we set the minimum support threshold to 0.1, as it manages to cover 48 courses with 1/16 of the itemsets. During the sampling step, we randomly select from 300 to 700 samples from either FI or MFI. The sampled course sets from FI might be shorter and less specific, allowing their easier activation from students' past histories. On the other hand, MFI will consist of the longest possible frequent sets of courses, so there will be no potential for overlap between the rules sampled. Statistics of the FI and MFI (with minimum support = 0.1) after sampling are presented in Table 2. We notice that for 700 samples, the average length of the course sets is similar for all the sets and the sampled ones. However, there are 5 and 2 fewer courses present in the FI and MFI samples, respectively. Next, for each course set, we ask ChatGPT4 to suggest $\{2, 3, 4, 5\}$ courses.

In the last step, using the rules, we generate recommendations and we explore different parameters that affect the process. We test the percentage of match threshold = [20, 30, 40, 50, 60, 70] to activate a rule based on the prior course history of each student when considering 1) all prior semesters, 2) the last two semesters, and 3) only the last semester. After that, we recommend new courses from 1) only the consequent part, and 2) the consequent and unmatched antecedent parts of the activated rule. For the third type of scoring rule that depends on matching ratio and sides of the activated rules, we have explored different values of α = [0, 0.001, 0.01, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45] and β = [0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6].

Course recommendation: For all the models, we recommend the courses with higher recommendation scores for the target semester. Following prior work [4, 2], we use a post-processing step where we remove any courses that the student has taken in the past or that are not offered for the target semester. Then, we recommend top k courses to each student where k is the number of courses the student wants to take in the target semester.

4.3. Evaluation Metrics

As in prior work [3, 4, 2], our primary evaluation metric is the **Recall@k** score, where k is the number of courses a student wants to take in the target semester. Recall computes the fraction of the target semester's courses that we recommended correctly. Since we recommend as many courses as a student plans to take in the target semester, recall and precision scores are equal in our case. Additionally, we compute the percentage of students to whom we can provide at least one relevant recommendation, **%1+ rel** [2]. Finally, we also present the number of instances for which students' history did not activate any rules, as in those cases, we can not generate personalized recommendations, so we just recommend the most popular courses.

4.4. Competing Approaches

We evaluate our models compared to other baselines and competing approaches that similarly use relevant data. For example, we do not consider methods that might use grading information. We particularly focus on comparing competing approaches that process the textual information in some form.

PopInTerm We implement a popularity-based approach as the baseline for course recommendation [44]. We use the students' course enrollment history available in training data to build this model. For each student, we start from the first semester and count how many students take a specific course in the first semester of their studies and do the counting for courses taken in the second semester and so on. The courses with the highest frequencies at the t-th semester are recommended for a student's t-th semester.

Association Rule Mining (ARM) As described in prior work [17, 18, 23], we implement an ARM approach, but we do not cluster the students based on their grades in prior courses, as we do not use grade information anywhere else. We consider each student's course history as one transaction. We implement the Apriori algorithm using scikit-learn [45] to generate high-confidence rules. For the recommendation part, we use the same process as described in our proposed approach, utilizing the confidence values generated by the Apriori algorithm for each rule. We explore minimum support = [0.10, 0.15, 0.20, 0.25] to generate itemsets, confidence threshold = [0.3, 0.4, 0.5, 0.6] to consider a rule, % of match threshold = [40, 50, 60, 70] to activate a rule matching with all prior courses of a target student. Similar to our work, we also explore recommending new courses from the consequent only and from both sides of an activated rule, which is not proposed in prior work.

TF-IDF We implement the TF-IDF model using the course descriptions for all courses to recommend similar and relevant courses [5, 6]. Each course description undergoes preprocessing, which involves converting text to lowercase, removing patterns and special characters, tokenizing the text into words, removing stopwords, and then lemmatizing the words to their base forms using *nltk* [46]. This step is critical for cleaning the text and reducing it to its essential content. Next, the course descriptions are transformed into numerical representations using TF-IDF, which captures the importance of each term within the course descriptions relative to their frequency across all descriptions. To compute the recommendation score of each course, we average over the cosine similarity between that course and prior courses taken in 1) all prior semesters, 2) the last two semesters, and 3) only the last semester of the target student.

PLANBERT We re-implement the PLANBERT model which is also a language model proposed in [7]. In this model, we do not use any future reference courses like the proposed model, only use the course enrollment data to train the model to capture historical course enrollment patterns. We make one sentence with all the courses taken by a student to prepare each instance of the training data. Then, we use the pre-trained DistilBERT model from the Hugging Face[5] and fine-tune it with our data. Using the data collator for Language Modeling[6], we tokenize the words including some masked tokens where α = [15, 20, 25] percent tokens (randomly chosen) are masked in each batch of data. Then we fine-tune the model with the training data where the masked tokens are the tokens to be predicted. To recommend, we use each student's previous courses to construct a single sentence and at the end of each sentence, we include a masked token. The fine-tuned PLANBERT model predicts and generates a score for each candidate course to be recommended for the upcoming semester. The courses with higher scores are recommended to each student.

5. Results

5.1. Performance Comparison

Table 3 presents the performance comparison between CRwLLM, CRwFine, GPTaCR models, and other existing approaches. Our proposed GPTaCR-FCS model (which uses the CNames+Desc context) outperforms the other approaches in all the metrics. It utilizes the power of ChatGPT for textual analysis. GPTaCR-FCS also performs better than the ARM and PLANBERT models for the test data providing slightly better recall (0.298) and %1+ rel scores (58.14%). ARM is already a well-established algorithm that captures all the associations among items present in the historical data since it considers all the high-confidence rules. In our approach, we partially and indirectly consider student enrollment data (only through the frequent course sets used). The method takes into account enrollment patterns. We can recommend the unmatched courses from the left-hand side of the rule, which includes the frequent course sets.

We observe a fine-tuned LLM model, CRwFine(llama) provides a better validation recall score than the GPTaCR model when we use only course enrollment data to re-train the base model. However, at the same time, the test performance is significantly lower, indicating that the fine-tuned recommendations do not generalize well for unseen instances. Interestingly, when we use both course enrollment and description data in the fine-tuning process, the performance of both fine-tuned models degrades. This happens because for each instance, we have much more text, and the model might have trouble focusing on the most relevant information.

Overall, while we partially use student enrolment data and provide limited course information, ChatGPT4 can build reasonable recommendations by taking advantage of its vast knowledge base. Fine-tuning an LLM is a potential solution to utilize all the course enrollment data by using them offline preserving the privacy of students but we might need more powerful LLMs to handle all the relevant information. The performance of our proposed models serves as a proof of concept and a first look into LLM's capabilities in recommendation tasks in education.

Table 3
Performance comparison of the proposed GPTaCR approach, fine-tuned LLMs, and competing approaches in terms of Recall@k and %1+ rel scores. CEnrol: Course enrollment, CDesc: Course descriptions, CNames: Course names, bothEN: CEnrol+CNames, bothND: CNames+CDesc, all: CEnrol+CNames+CDesc.

Model	Data used	Recall@k		%1+ rel
		Valid	Test	
PopInTerm	CEnrol	0.160	0.104	26.21
TFIDF	CDesc	0.047	0.058	15.65
PLANBERT	CEnrol	0.341	0.273	55.60
ARM	CEnrol	**0.358**	0.292	56.31
CRwLLM	bothEN	0.049	0.043	11.83
CRwFine(Llama)	CEnrol	0.354	0.266	52.74
CRwFine(Llama)	all	0.323	0.250	51.87
CRwFine(GPT)	CEnrol	0.345	0.258	51.87
CRwFine(GPT)	all	0.339	0.254	48.93
GPTaCR-1C	bothND	0.151	0.135	33.36
GPTaCR-FCS	all	0.342	**0.298**	**58.14**

Table 4
Effects of different hyperparameters in GPTaCR-1C model. Here, RHS = right-hand side of a rule. The second row corresponds to the best overall model, while the other rows have the best model for the listed parameter.

RHS size	History used	Scoring rule	Recall@k		%1+ rel
			Valid	Test	
4	**all**	**course-taking semester**	0.151	**0.135**	**33.36**
5			0.134	0.100	24.15
3			0.101	0.100	24.63
	last 2		**0.155**	0.133	32.65
	last 1		0.143	0.113	27.40
		baseline-counting	**0.155**	0.133	32.65

5.2. Effects of Hyperparameters in GPTaCR

Since the GPTaCR models perform the best, we further delve into how they are affected by different hyperparameters. We present the effect of different hyperparameters of GPTaCR-1C model in Table 4. We observe best test Recall@k and %1+ rel scores by asking ChatGPT4 to suggest 4 courses,

[5]https://huggingface.co/learn/nlpcourse/chapter7/3?fw=tf
[6]https://huggingface.co/docs/transformers/main_classes/data_collator#transformers.default_data_collator

Table 5
Effects of different hyperparameters in GPTaCR-FCS model. Here, RHS = suggested courses from ChatGPT in the consequent part of a rule. The second row corresponds to the best overall model, while the other rows have the best model for the listed parameter.

Provided Context	Sample size	RHS size	% of match	History used	Rules' side used	Scoring	Recall@k		%1+ rel	Cases not covered	
							Valid	Test		Valid	Test
CNames+ Desc	**600**	**3**	**30**	**all**	**both**	**matching ratio**	0.342	**0.298**	**58.14**	12.7	2.0
CNames							0.332	0.290	56.90	16.0	3.7
	700						**0.343**	0.293	55.53	12.3	2.0
	500						0.336	0.294	57.72	15.7	2.7
	400						0.340	0.294	57.61	22.7	4.3
	300						0.332	0.296	57.72	32.0	6.0
		5					0.338	0.290	57.29	14.0	2.0
		4					0.331	0.292	56.9	57.3	9.0
		2					0.336	0.298	57.82	19.3	3.3
			20				0.340	0.288	56.45	31.7	23.0
			40				0.338	0.294	57.53	36.7	7.3
			50				0.329	0.289	56.74	82.0	13.0
			60				0.296	0.269	54.01	227.0	24.7
			70				0.280	0.250	51.92	332.3	50.0
				last 2 sem			0.340	0.288	56.45	31.7	23.0
				last 1 sem			0.303	0.240	49.11	206.0	156.0
					RHS		0.086	0.101	24.91	426.0	65.7
						ChatGPT	0.335	0.295	57.61	14.3	3.0
						baseline-counting	0.337	0.294	57.61	12.3	2.0

using all the history of a student to match with the one prior course (LHS) of a rule and scoring based on course-taking semester. We observe the biggest variability for the size of the right-hand side, while the scoring method does not dramatically change the results.

We present the effects of different hyperparameters of the GPTaCR-FCS model in Table 5. In the second row, we present the best set of hyperparameters for whom we get the best Recall@k and %1+ rel scores. The performance observed refers to the case when we use FI samples rather than MFI samples. In the latter case, we get test recall 0.277 and %1+ rel 54.83%. One reason is that students manage to activate more rules in FI samples rather than in MFI, resulting in more reliable recommendations in the end. Moving forward, we will only present results and discuss approaches that sample from FI.

In the other rows of Table 5, we present the best scores for each parameter value where empty cells indicate the best possible hyperparameters for the parameter noted in that row. First, we can see that we get the best scores when we use more contexts (course names, acronyms, and descriptions) than only using course names. The improvement in performance though is quite limited, indicating that ChatGPT4 uses its existing knowledge to infer the topics covered by only using the course names. As a result, when we add the course descriptions, we do not get a significant improvement.

Second, we observe better performance when we use more samples (best validation recall using 700 samples and best test recall using 600 samples). It is understandable that if we use fewer samples (and consequently, fewer rules), we get more instances with few or no activated rules, leading to unreliable recommendations. Third, we get the best scores with 3 courses suggested by ChatGPT4, while the other values have similar performance. A total of 3 courses in a semester are also commonly taken by the students of our dataset. When we ask ChatGPT4 to suggest more than 3 courses, it may suggest some less relevant courses and overall recall scores become lower.

Fourth, we can see better performance when using the lower percentage of thresholds (20, 30, 40) of matching the antecedents of the rules with the students' prior courses. One possible reason could be that the more we increase the match threshold, the fewer rules activated, resulting in diminished recommendation performance. Fifth, we also examine the best performance achieved when we consider all, the last 2, or only the last semester of a student's registration history to match it to the rules. The results indicate that the more history we use, the better results we will get. When we use all history, we manage to match more rules, resulting in better performance and a lower number of instances for which we cannot offer personalized recommendations. Sixth, we get much better results when we use both sides of the activated rules to recommend courses to a student, i.e., both the consequent courses and any unmatched courses in the antecedent that the student has not taken yet. This indicates that the frequent course sets capture important historical trends in the enrolment data about which courses are commonly taken together. This is an indirect way of taking into account student-course interactions.

5.3. Discussion and Limitations

We need to acknowledge some limitations existing when using ChatGPT to build a recommendation system for university students. Firstly, we need to be very careful when using sensitive student data. As mentioned before, we cannot directly put real-world student-course interactions to ChatGPT4 due to privacy concerns about students' data. Secondly, it might be tricky to track what part of the provided

information ChatGPT4 uses to generate the recommendation. These tools have a particular limit on the length of text that they can process, and it can be hard to track this when interacting with the tool. If someone is not careful, ChatGPT might return suggestions while forgetting information provided earlier, resulting in meaningless output. That is even more common when using ChatGPT3.5. Additionally, LLMs suffer from hallucinations sometimes, which in our case could mean suggesting invalid course codes. In our experiments, we noticed this phenomenon only when our prompts had too long input character length. Once we took care of that, we checked but did not notice any hallucinations again. In general, someone should validate the output to ensure the outcomes are meaningful.

Third, for the confidence scores provided by ChatGPT4, we do not have a clear idea about how they are calculated, so they are not very reliable, as shown by our results (since the other scoring methods perform better). Finally, LLMs do not offer reliable recommendations when provided with just course content and one student's registration history. This serves as a warning for the students who might try to directly use an LLM with their own data.

6. Conclusion

Our goal is to explore ways to utilize LLMs for the task of course recommendation. ChatGPT4 can understand conversational text data and analyze the provided context to suggest relevant courses. We evaluate direct methods for recommendation, where we directly offer information to the model and request recommendations for the next courses a student should take. We also introduce a novel framework that uses ChatGPT4 to create rules between prior and future courses. We also explore providing different contexts (course names, acronyms, and course descriptions) to ChatGPT4 so that it can find relevant and related courses. Our results highlight that when using directly ChatGPT4, we might not have relevant recommendations, but our proposed approach manages to better capture semantic information on course information and the insights provided by historical enrollment data. Based on our work, we believe more research is needed to explore the LLM capabilities in course recommendation. We hope to inspire other EDM researchers to consider LLMs for different problems in educational settings.

References

[1] A. Kadlec, J. Immerwahr, J. Gupta, Guided pathways to student success perspectives from indiana college students and advisors, New York: Public Agenda (2014).

[2] A. Polyzou, A. N. Nikolakopoulos, G. Karypis, Scholars walk: A markov chain framework for course recommendation., International Educational Data Mining Society (2019).

[3] M. A. Z. Khan, A. Polyzou, Session-based course recommendation frameworks using deep learning, in: Proceedings of the 16th International Conference on Educational Data Mining, 2023, pp. 269–277.

[4] Z. A. Pardos, Z. Fan, W. Jiang, Connectionist recommendation in the wild: on the utility and scrutability of neural networks for personalized course guidance, User modeling and user-adapted interaction 29 (2019) 487–525.

[5] J. Naren, M. Z. Banu, S. Lohavani, Recommendation system for students' course selection, in: Smart Systems and IoT: Innovations in Computing, Springer, 2020, pp. 825–834.

[6] Z. A. Pardos, W. Jiang, Designing for serendipity in a university course recommendation system, in: Proceedings of the tenth international conference on learning analytics & knowledge, 2020, pp. 350–359.

[7] E. Shao, S. Guo, Z. A. Pardos, Degree planning with plan-bert: Multi-semester recommendation using future courses of interest, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 14920–14929.

[8] E. Revell, Nearly half of job seekers use AI to polish their resumes, finance.yahoo.com, 2024. https://finance.yahoo.com/news/nearly-half-job-seekers-us-110254846.html.

[9] C. Edwards, Teaching Transformed, Communications of the ACM, 2024. https://cacm.acm.org/magazines/2024/2/279522-teaching-transformed/fulltext.

[10] Realize the business value of generative AI in your organization, Amazon Web Services, 2024. https://aws.amazon.com/generative-ai/use-cases/.

[11] J. Liu, C. Liu, R. Lv, K. Zhou, Y. Zhang, Is chatgpt a good recommender? a preliminary study, arXiv preprint arXiv:2304.10149 (2023).

[12] J. Liu, C. Liu, P. Zhou, Q. Ye, D. Chong, K. Zhou, Y. Xie, Y. Cao, S. Wang, C. You, et al., Llmrec: Benchmarking large language models on recommendation task, arXiv preprint arXiv:2308.12241 (2023).

[13] S. Bhandari, Y. Liu, Z. A. Pardos, Evaluating chatgpt-generated textbook questions using irt, ????

[14] R. Yilmaz, F. G. K. Yilmaz, Augmented intelligence in programming learning: Examining student views on the use of chatgpt for programming learning, Computers in Human Behavior: Artificial Humans 1 (2023) 100005.

[15] N. M. S. Surameery, M. Y. Shakor, Use chat gpt to solve programming bugs, International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290 3 (2023) 17–22.

[16] K. Lekan, Z. A. Pardos, Ai-augmented advising: A comparative study of chatgpt-4 and advisor-based major recommendations, ????

[17] A. Al-Badarenah, J. Alsakran, An automated recommender system for course selection, International Journal of Advanced Computer Science and Applications 7 (2016) 166–175.

[18] N. Bendakir, E. Aïmeur, Using association rules for course recommendation, in: Proceedings of the AAAI workshop on educational data mining, volume 3, Citeseer, 2006, pp. 1–10.

[19] A. Esteban, A. Zafra, C. Romero, A hybrid multicriteria approach using a genetic algorithm for recommending courses to university students., International educational data mining society (2018).

[20] B. Ma, M. Lu, Y. Taniguchi, S. Konomi, Exploration and explanation: An interactive course recommendation system for university environments, in: CEUR Workshop Proceedings, volume 2903, CEUR-WS, 2021.

[21] M. S. Sulaiman, A. A. Tamizi, M. R. Shamsudin, A. Azmi, Course recommendation system using fuzzy logic approach, Indonesian Journal of Electrical Engineering and Computer Science 17 (2020) 365–371.

[22] C. De Medio, C. Limongelli, F. Sciarrone, M. Temperini,

Moodlerec: A recommendation system for creating courses using the moodle e-learning platform, Computers in Human Behavior 104 (2020) 106168.
[23] R. Obeidat, R. Duwairi, A. Al-Aiad, A collaborative recommendation system for online courses recommendations, in: 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML), IEEE, 2019, pp. 49–54.
[24] H. Zhang, T. Huang, Z. Lv, S. Liu, Z. Zhou, Mcrs: A course recommendation system for moocs, Multimedia Tools and Applications 77 (2018) 7051–7069.
[25] P. Symeonidis, D. Malakoudis, Multi-modal matrix factorization with side information for recommending massive open online courses, Expert Systems with Applications 118 (2019) 261–271.
[26] E. Bauer, M. Greisel, I. Kuznetsov, M. Berndt, I. Kollar, M. Dresel, M. R. Fischer, F. Fischer, Using natural language processing to support peer-feedback in the age of artificial intelligence: A cross-disciplinary framework and a research agenda, British Journal of Educational Technology (2023).
[27] W. Dai, J. Lin, H. Jin, T. Li, Y.-S. Tsai, D. Gašević, G. Chen, Can large language models provide feedback to students? a case study on chatgpt, in: 2023 IEEE International Conference on Advanced Learning Technologies (ICALT), IEEE, 2023, pp. 323–325.
[28] Y. Ehara, Measuring similarity between manual course concepts and chatgpt-generated course concepts (2023).
[29] Z. A. Pardos, S. Bhandari, Learning gain differences between chatgpt and human tutor generated algebra hints, arXiv preprint arXiv:2302.06871 (2023).
[30] A. Botelho, S. Baral, J. A. Erickson, P. Benachamardi, N. T. Heffernan, Leveraging natural language processing to support automated assessment and feedback for student open responses in mathematics, Journal of Computer Assisted Learning (2023).
[31] G. Pinto, I. Cardoso-Pereira, D. Monteiro, D. Lucena, A. Souza, K. Gama, Large language models for education: Grading open-ended questions using chatgpt, in: Proceedings of the XXXVII Brazilian Symposium on Software Engineering, 2023, pp. 293–302.
[32] T. Phung, J. Cambronero, S. Gulwani, T. Kohn, R. Majumdar, A. Singla, G. Soares, Generating high-precision feedback for programming syntax errors using large language models, arXiv preprint arXiv:2302.04662 (2023).
[33] D. Wang, D. Shan, Y. Zheng, K. Guo, G. Chen, Y. Lu, Can chatgpt detect student talk moves in classroom discourse? a preliminary comparison with bert, in: Proceedings of the 16th International Conference on Educational Data Mining, International Educational Data Mining Society, 2023, pp. 515–519.
[34] J. M. Markel, S. G. Opferman, J. A. Landay, C. Piech, Gpteach: Interactive ta training with gpt based students (2023).
[35] C.-Y. Chang, G.-J. Hwang, M.-L. Gau, Promoting students' learning achievement and self-efficacy: A mobile chatbot approach for nursing training, British Journal of Educational Technology 53 (2022) 171–188.
[36] T. Shaik, X. Tao, C. Dann, H. Xie, Y. Li, L. Galligan, Sentiment analysis and opinion mining on educational data: A survey, Natural Language Processing Journal 2 (2023) 100003.
[37] L. Yan, L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, X. Li, Y. Jin, D. Gašević, Practical and ethical challenges of large language models in education: A systematic literature review, arXiv preprint arXiv:2303.13379 (2023).
[38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners (2019).
[39] AI@Meta, Llama 3 model card (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
[40] F. Jia, K. Wang, Y. Zheng, D. Cao, Y. Liu, Gpt4mts: Prompt-based large language model for multimodal time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 23343–23351.
[41] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, Advances in Neural Information Processing Systems 36 (2024).
[42] OpenAI, Chatgpt-4, 2023. URL: https://chat.openai.com, software available from OpenAI.
[43] S. Raschka, Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack, The Journal of Open Source Software 3 (2018). URL: https://joss.theoj.org/papers/10.21105/joss.00638. doi:10.21105/joss.00638.
[44] A. Elbadrawy, G. Karypis, Domain-aware grade prediction and top-n course recommendation, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 183–190.
[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
[46] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.