Interactive Rubric Generator for Instructor's Assessment Using Prompt Engineering and Large Language Models

Haoze Du
North Carolina State University
Raleigh, NC, USA
hdu5@ncsu.edu

Parvez Rashid
North Carolina State University
Raleigh, NC, USA
mrashid4@ncsu.edu

Qinjin Jia
North Carolina State University
Raleigh, NC, USA
qjia3@ncsu.edu

Edward Gehringer
North Carolina State University
Raleigh, NC, USA
efg@ncsu.edu

***Abstract*—This research paper describes an interactive system using large language models and prompt engineering to generate rubrics. Rubrics have long been employed to ensure a grading system that is both equitable and consistent. In practice, generating rubrics could be challenging for instructors for many reasons (e.g., a tight course schedule, limited resources, and varying materials for different projects in the same course), which urges the need to generate the rubric automatically. To the best of our knowledge, little research has been performed on generating rubrics. In this work, we present a novel system based on Large Language Models (LLMs) and Prompt Engineering to help instructors generate rubric items interactively based on course materials, as well as assess the student's work using these rubrics to give timely feedback automatically. In this system, we applied several LLMs (e.g., GPT4, Llama, Falcon, and Hermes) to generate both rubric and feedback using this process: 1) a set of text chunks are initially generated from the textual materials (these textual materials may from various sources), then LDA (Latent Dirichlet Allocation) is applied to extract a set of keywords from the preprocessed text chunks for rubric generation; 2) a web page was designed to let the instructor choose if the keywords from the set are adequate as rubric words; 3) the rubric items are generated by LLMs from the rubric words. In our experiments, a total number of 1017 documents (including the syllabus, the course website, the requirement of projects, the students' works, and the instructors' feedback) were used to build the corpus to generate the rubric-related keywords. Three users (including one instructor and two teaching assistants) participated in generating the rubric interactively using the webpage. The results of experiments show that the interactively generated rubrics from the LLM-instructor system can achieve a level similar to manually created rubrics. We utilized different prompts to let the LLMs generate feedback for the student's work, based on the generated rubrics. Our study shows that generating automatic rubrics and feedback for student project reports is feasible, yet it also identifies significant challenges that future research needs to address.**

***Index Terms*—Large Language Models (LLMs), Automatic Rubric Generation, Latent Dirichlet Allocation (LDA), Feedback Automation.**

I. Introduction

The design of course projects is a crucial element of university STEM courses, particularly in software development. These projects bolster students' theoretical knowledge and enhance their practical application skills within the field [1], [2]. A significant aspect of these courses involves the assessment of project reports submitted by students. To ensure a fair and objective evaluation, it is vital to develop rubrics that delineate the criteria for grading. However, these reports are usually written and assessed very subjectively. So it is vital to have such rubrics to guide the assessment process, ensuring that all students are evaluated against the same standards, essential for maintaining the integrity and consistency of educational evaluations [3]. Moreover, well-defined rubrics contribute to the quality of education by providing students with transparent expectations and benchmarks, which are crucial for their academic and professional development [4]. Thus, exploring innovative methods to generate and refine assessment rubrics is essential and is emerging as a critical area of focus in educational research.

With the advancement of artificial intelligence technologies, transformer-based large-scale language models, such as Transformer-based Bidirectional Encoder Representations (BERT) [5] and Generative Pre-trained Transformers (GPT) [6], have demonstrated near-human-level performance in various language tasks [7]. These models have shown remarkable performance in multiple domains, including AI tasks and code generation [8]. They have also found applications in the field of education, where Jia et al. [9] proposed a data-driven and BART-based method for generating automatic feedback on student project reports. The process was evaluated using a series of metrics to measure the similarity between the generated feedback and human evaluation, demonstrating the feasibility of automatically generating feedback for student

project reports. Moore et al. [10] proposed an annual workshop focused on leveraging AI and learning analytics for content generation in education to explore new methods to create and assess educational content.

This study proposes an innovative approach to generating rubrics for student assignments using LLMs and evaluates the quality of generated rubrics in multiple dimensions. The existing evaluation reports are used in carefully designed prompts to ensure that the feedback generated by the AI has content and structure similar to that of human experts. Multiple paragraphs are generated for each prompt, and the final paragraph is selected based on a scoring agreement with the original human expert evaluations. The selected paragraphs are manually edited to ensure accuracy, correcting any grammar and factual errors. To evaluate the effectiveness of the AI-generated feedback evaluations, human experts assess the generated results based on five metrics: Readability, Suggestions, Problems, Positive Tone, and Factuality. The effectiveness of the design method is validated through comparative experiments with different LLM models.

In this paper, we introduce the Rubric Generator (RG) framework, an innovative system that integrates keyword extraction, prompt engineering, and the capabilities of a large language model (LLM). This comprehensive framework is designed to process textual materials as input and systematically generate a detailed rubric. Furthermore, the RG framework is engineered to leverage the generated rubric to produce feedback automatically. This automated feedback mechanism is a pivotal feature, facilitating a seamless and efficient evaluation process that can significantly enhance the instructional workflow and improve educational outcomes. Through its dual functionality, the RG framework streamlines rubric creation and ensures that the feedback provided is relevant and contextually appropriate.

The remaining parts of the paper are organized as follows: Section II introduces the previously mentioned methods to generate textual materials, especially in education field; Section III describes the process we applied for this task; Section IV describes the experiments and comparisons we made to evaluate the performance of the proposed method Rubric Generator (RG); Section V analyzes the results and compares the performances. Sections VI present the conclusion, the limitations of our current work, and the future work we plan to do to make some improvements.

II. Related Works

A. Large language models(LLM)

The development of large models in natural language processing can be traced back to 2018 when Google introduced BERT. The release of BERT garnered significant attention and application, as it achieved leading performance in various natural language processing tasks such as text classification, named entity recognition, sentiment analysis, and more. BERT's success demonstrated the potential of large models in natural language processing and spurred further research and development. Following BERT, several large models emerged, including GPT (Generative Pre-trained Transformer), XLNet, RoBERTa, and others. These models built upon BERT and introduced improvements and optimizations, enhancing their performance in natural language processing tasks.

As large models continued to evolve, researchers started exploring ways to make them even larger and more powerful. In 2020, OpenAI released a series of large models, "Generative Pre-trained Transformers" (GPT), that excel in few-shot learning, meaning they can perform tasks with a small number of representative text samples without pre-training. The newest version is named GPT-4 [11], released in 2023. GPT-4 showcased impressive capabilities in various natural language processing tasks but also posed challenges regarding computational resources and training time. In addition to scaling up model size, researchers have also focused on improving the efficiency and interpretability of large models. Techniques such as model compression, parameter sharing, and knowledge distillation have been explored to reduce LLMs' computational and storage requirements, making them more suitable for practical applications.

The development of these models and their variants has sparked new research on their potential applications in various fields. In education, it has led to a surge in application development.

B. Applications of large language models in education

As large language models (LLMs) continue to advance, many researchers are exploring their applications in education. Rahman and Watanobe [12] investigated the impact of ChatGPT (a variant of the GPT model) on higher education, with a specific focus on computer programming subjects. The evidence provided by the authors emphasizes the effectiveness of ChatGPT in managing programming assignments, exams, and homework. The GPT model and its continuous iterations have also been used in educational research, such as automatic question generation (e.g., [13], [14]), automated grading (e.g., [15], [16]) and generating educational content (e.g., [17]).

C. Prompt engineering in LLM

Prompt engineering is an essential technique that guides language models to generate more accurate, coherent, and specific outputs by adjusting and optimizing the input prompts. This technique has wide-ranging applications and potential in academic research, natural language processing, and artificial intelligence. By designing and optimizing prompts effectively, the quality and accuracy of model outputs can be improved, enabling them to meet specific requirements and application scenarios better. However, prompt engineering also requires developers to possess domain knowledge and skills to ensure the models generate the expected results.

D. Latent Dirichlet Allocation (LDA)

LDA [18], proposed in 2003, is used to infer the topic distribution of documents. It can provide each document's topic distribution in a document set as a probability distribution. We can perform topic clustering or text classification

based on the topic distribution by analyzing some documents and extracting their topic distributions. LDA adopts the bag-of-words model, which considers only the occurrence of vocabulary in a document without considering the order of occurrence. For the evaluation task of student project feedback, we need to determine the design based on the course syllabus and the requirements of the course design project, focusing on the five dimensions of human expert evaluation criteria. Here, we primarily use LDA to rank the relevant keywords $r(w, k \mid \lambda)$, and the definition of Relevance is as follows[19]:

In this study, we define the relevance of term w to topic k given a weight parameter λ (where $0 \leq \lambda \leq 1$) as:

$$r(w, k \mid \lambda) = \lambda \log(\phi_{kw}) + (1 - \lambda) \log\left(\frac{\phi_{kw}}{p_w}\right) \tag{II.1}$$

Where λ determines the weight given to the probability of term w under topic k relative to its lift (measuring both on the log scale). ϕ_{kw} denotes the probability of term $w \in 1, ..., V$ for topic $k \in 1, ..., K$, where V denotes the number of terms in the vocabulary, and p_w denotes the marginal probability of term w in the corpus. One typically estimates ϕ in LDA using Variational Bayes methods or Collapsed Gibbs Sampling and p_w from the empirical distribution of the corpus (optionally smoothed by including prior weights as pseudo-counts).

The formula for Collapsed Gibbs Sampling is as follows:

$$\begin{aligned} p_w &= p(w|z, \theta, \phi_{kw}) \\ &= \frac{p(z, w|\theta, \phi_{kw})}{p(z|\theta, \phi_{kw})} \\ &= \frac{p(w|z, \phi_{kw})p(z|\theta)}{\sum_{w'} p(w'|z, \phi_{kw})p(z|\theta)} \end{aligned} \tag{II.2}$$

where, $p(w|z, \theta, \phi_{kw})$ represent the probability of word w given topic z and the current topic-word distribution and document-topic distribution; $p(z|\theta)$ represents the probability of topic z given the document-topic distribution; $p(w|z, \phi_{kw})$ represents the probability of word w given topic z and the topic-word distribution; θ represents the document-topic distribution, and ϕ_{kw} represents the topic-word distribution. The formula for Collapsed Gibbs Sampling is as follows:

1) Initialize the topic-word distribution and document-topic distribution.
2) For each word w in each document d:
   a) Sample a topic z from the topic-word distribution.
   b) Update the document-topic distribution and topic-word distribution.
3) Repeat step 2 until convergence criteria are met.

Collapsed Gibbs Sampling can estimate the parameters in the LDA model by iterative sampling, resulting in the topic-word and document-topic distributions.

III. Methods

This section introduces the design of the model to generate the rubric. It includes the construction method of prompt engineering corpus, acquisition of training sets, rubric keyword lists, interactive selection of grading criteria keywords by teachers and scoring for each grading criterion keyword, and the implementation process of generating rubric items using an LLM.

A. Problem Description

The mathematical model for the rubric generation problem can be defined as follows: Given some textual materials X (the textbook, the syllabus, etc.) and the materials Y which are related to the specific assignment (the description of the assignment, the student's submissions, etc.), the task is to generate a rubric R. To formalize this, we can define it as:

$$Rubric = RG(X, Y) \tag{III.1}$$

Where RG is the rubric generator we proposed.

B. Overall Structure of Rubric Generation

Based on the analysis of student project feedback requirements, this paper uses the framework to complete the automatic feedback system RG design for student project reports by designing the corresponding information pipeline and fine-tuning. Figure 1 shows the specific research process.

Figure 1 illustrates three main phases to generate the rubric with the given assignment materials.

The steps below show how we construct the rubric generator RG we proposed for technically implementing Phase 1 and 2:

1) Build a corpus from the textual materials using LDA to derive a keyword corpus $R_X = LDA(X)$ with a relevance score λ_i for rubric keyword $r_i \in R_X$.
2) Extract the keywords $R_Y = LDA(Y)$ from the assignment based materials Y using LDA. Then, generate a rubric keywords list $R = R_X \cap R_Y$. Drop any rubric keyword r_i if $\lambda_i < \theta_X$, here θ_X is a threshold and is considered a hyperparameter in this model.
3) Interactively let the instructor choose the rubric keywords from the list R. Add more rubric keywords manually and assign the points p to each rubric keyword to grade the student's assignment.
4) Encapsulate the interactively selected rubric keywords as a list with the given points denoted as $RL = \{(r, p), \cdots\}$. Then utilize LLM to generate the detailed rubric terms based on RL with the prompts.

IV. Implementation and Experiments

This section describes the implementation and the experiments we have done in this work. All codes, including the interfaces and backend, are released in a GitHub repository[1].

A. Data Source

The data used in this work were collected from the graduate-level object-oriented software development class at our university, which uses the Expertiza [20] peer-assessment platform. In this class, students are required to develop, refactor, and/or test new code for Expertiza, and then document what they have

[1]https://anonymous.4open.science/r/RubricGenerator-6B4B/

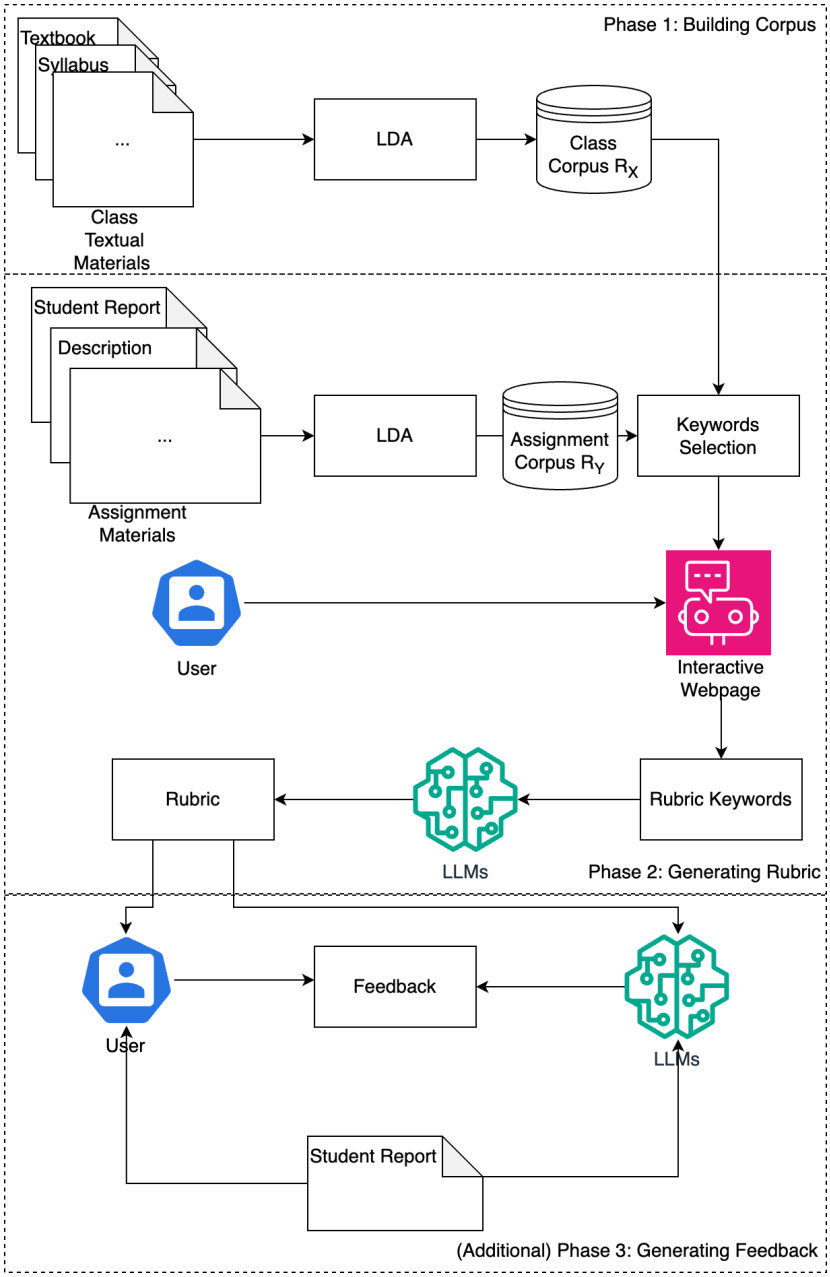

Fig. 1. General procedure of RG. Phase 1: Build the class corpus R_X from textual materials; Phase 2: extract keywords from assignment materials using LDA and select the keywords by intersecting the similar words from the corpus R_X. Let the user choose interactively the rubric keywords. Finally, use the LLM to generate a rubric; Phase 3 (additional): Generate feedback with the rubric from the previous phase.

developed.[2] The instructor reviews these reports and provides textual feedback on each report. Table I shows the instructor's prose reviews for randomly chosen students' reports (without using the generated rubric). We gathered project reports from semesters between Spring 2015 and Spring 2023, to generate rubrics for the given assignments.

[2]Sample student reports can be seen at https://anonymous.4open.science/r/RGF-EF55/

B. Data Preprocessing

The course materials such as the textbook, the syllabus, the assignment descriptions are distributed as PDF files. We utilize OCR to process the original PDF file, to extract the textual data for the following steps. Student reports is formatted as a Wiki page, in a JSON file. So, (1) we extract the text from the raw data with labels; (2) tokenize each extracted report; (3) then split the tokenized extracted text into chunks to make the number of tokens in each chunk not exceed the limit for the relevant LLM. A sequential label is also attached to the head of each chunk, to ensure the completeness of the data when

TABLE I
ORIGINAL INSTRUCTOR'S REVIEWS FOR RANDOMLY SELECTED REPORTS.

Index	Instructor review
1	It is difficult to follow the changes for Problem 1. The problem description is separated from the code snippet. The new code is inserted in the code shown above, but it's not clear where the new code resides. A Github diff view would have been much clearer. This is also true for Problem 2. The first change to tests needs to be described more fully. The description of the second test is quite confusing and should be reworded.
2	The documentation does not mention that they skipped Issue #256. However, it is nicely structured thus it's easy to find information. However the solution presented are mostly just copy paste of the code. It should be described in a narrative, rather than a list of bullet points.
3	Good writeup, and good motivation for the changes that you made. Would have been helpful to explain the magic hex constants in the tests.

connecting to the LLMs. The instructor feedback is collected from the back end of the Expertiza platform, in CSV format. We randomly split the assignments and the feedback data into two sets, as Table II shows.

TABLE II
THE FEEDBACK DATASET

Feedback set	Size	Explanation
Corpus set (X)	(79.9%, 885)	The data to build the corpus.
Test set (Y)	(20.1%, 222)	The data for generating rubrics only.

C. Interactively Generating Rubric Keywords

Langchain [21] is used to build the framework of the entire implementation, to run the LLMs in parallel and compare them. In this work, the keyword corpus R_X (defined in Section III-B) is collected from the textual materials, including the textbook, the syllabus, the student's assignments, and related instructors' feedback from the corpus data set (and not from the test set). The materials specific to the assignment referred to as Y, are configured to allow users to upload, to generate the extracted keywords list R_Y. The webpage contains checkboxes and a text area for supplementary rubric keywords, enabling users to select rubric keywords interactively, as depicted in Figure 2.

D. Generating Rubric from Keywords

An LLM prompt is essential for generating a rubric based on specified rubric keywords. For example, the prompt outlined below is designed to create a rubric specifically for a student report assignment:

```
"Your task is to generate a rubric from the
given keywords and related points. Please
generate the rubric formatted as below:
| Term | Points | Description |.
And the following input string
contains the keywords and related
points, formatted as (keywords, points)"
```

Using this prompt, the LLM can generate a detailed rubric to help instructors provide targeted feedback that is directly relevant to the assignment's objectives and the student's performance.

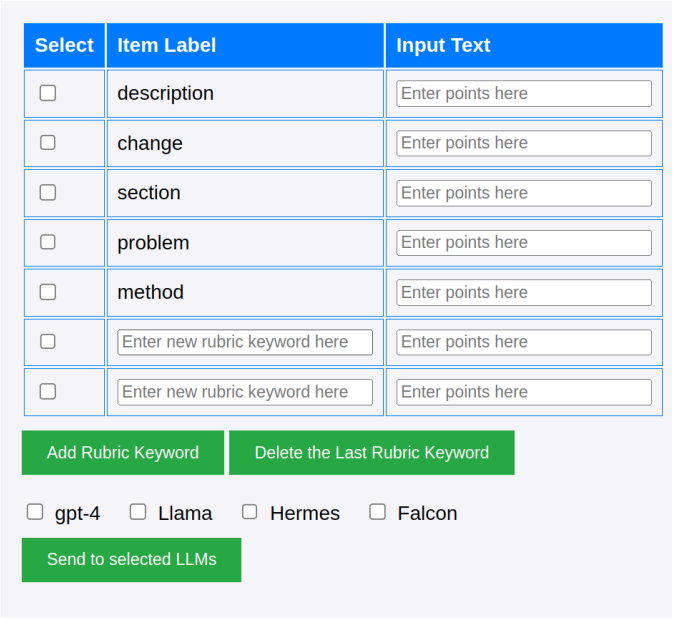

Fig. 2. The interactive user interface to select rubric keywords

To compare the quality of the generated rubrics by different LLMs, we created a Langchain[21] pipeline as the basic framework, as Figure 3 shows. The table III lists the LLMs we use to generate the rubric from the textual materials.

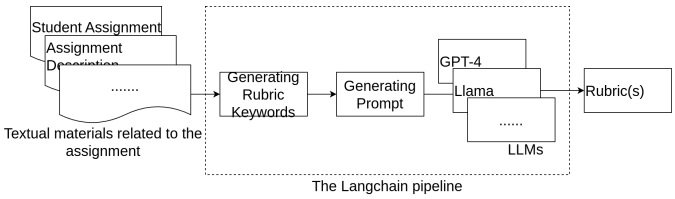

Fig. 3. The Langchain framework for rubric generation.

TABLE III
LLMS FOR GENERATING RUBRICS

LLMs	Size	Description
GPT-4 [11]	~45GB	Remote LLM provided by OpenAI, the size is from their document.
Llama-3 [22]	4.84GB	Local LLM implemented by Meta
Falcon [23]	3.92GB	Local LLM implemented by Nomic-AI
Hermes [24]	3.83GB	Local LLM implemented by Nous

E. Generating Feedback from the Rubrics

Additionally, the Rubric Generator (RG) can utilize LLMs with interactively generated rubrics to enhance the feedback process by generating feedback for the student. The resulting in feedback is considerably more structured compared to methods that rely solely on feedback generated from a language model. By incorporating a rubric, the feedback becomes anchored to specific assessment criteria, offering more straightforward guidance and measurable benchmarks

for student improvement. This structured approach facilitates a more transparent and effective evaluation process, helping students understand their performance.

We have developed an interface enabling instructors to automatically generate feedback using LLMs, as illustrated in Figure 4, which takes an assignment and generates feedback as output. The feedback produced using the rubric generated from language models is evaluated by comparing it with the original feedback collected from the test set and with feedback derived from the generated rubric. This comparison helps assess the effectiveness of both the generated rubric and the feedback. Since this is not the main focus of this paper, we have included the prompt used to generate the feedback in Appendix A.

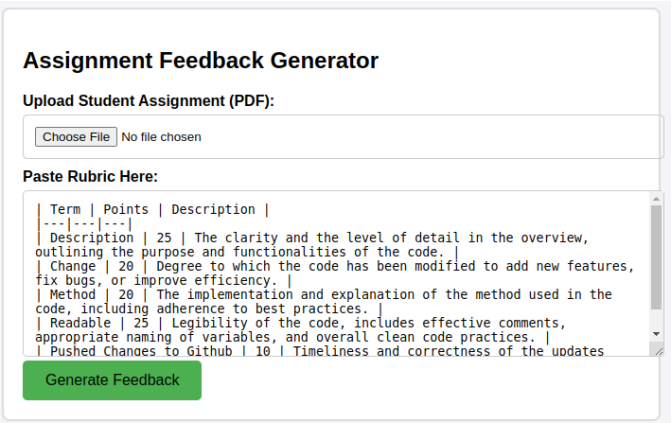

Fig. 4. The interface for generating feedback.

V. Results and Comparison

In this section, we present the results and discuss the findings from the generated rubric and feedback. One example (from OpenAI GPT-4) has been randomly selected from our test set and presented in Table IV. We perform the interactive experiment on 20 assignments, including submissions and reviews, to evaluate the quality of the generated rubric and the related feedback.

A. Interactive Rubric Analysis

As demonstrated in Table IV, the interactively generated rubric exhibits a well-organized structure for each item. In Table V, we manually compare the quality of the rubric with/without interaction between users by human evaluations from the following fields from the Jia et al. [9]: the readability, clearness, and conciseness (scale from 0–10) are evaluated subjectively by a set of users (including one instructor and two teaching assistants). The number of rubric keywords is set as 5 for each input, and the users are allowed to delete improper rubric keywords or add new rubric keywords. Additionally, to compare the quality of the generated rubrics without the interactively selecting rubric keyword step, we directly input the assignment documents (description of assignment, student assignment, feedback) to LLM to get the rubric as a control group. The enhanced logic and understanding of related terms are due to the interactive involvement of a human user, which improves the overall quality of the rubric, as Table V shows.

B. Rubric from Different LLMs

We conduct another manual rubrics assessment for different LLMs, employing consistent subjective metrics on the same input assignment materials generated through interactive steps with different LLMs, as shown in Table VI. GPT-4 delivers the highest quality in generated materials. This superior performance might be related to the size of the LLM, which is approximately 45GB, as indicated in Table III. Another possible reason for GPT-4's superior performance could be that it is a more finely tuned language model than the other three, demonstrating optimal quality for generating rubrics. It suggests that fine-tuning should be applied, especially for tasks such as generating rubrics from the given textual materials.

The results further indicate that commercial large language models (LLMs) like GPT-4 tend to outperform their open-source counterparts in terms of performance. However, local LLMs offer enhanced privacy benefits, in contrast to remote models, which carry the risk of data leakage. This highlights a trade-off between performance and privacy when choosing between different types of LLMs for specific applications.

C. Feedback from the Generated Rubric

This part presents the results showing the feedback automatically generated from the rubric, as described in Section IV-E. Here, we contrast our results with (1) the automatically generated feedback from Jia et al. [9], (2) directly let LLM generate feedback and (3) the original feedback provided by the instructor from the test set, as detailed in Table VII.

In TableVII, the dimensions of readability, suggestions, and tone are selected to evaluate the feedback manually after reviewing related works [9], [25], [26], and "Rubric?" "Grade?" indicates whether the method can generate a rubric or an overall grade for a given student assignment. From the comparative analysis of different methods, it is evident that our proposed method (LLM + RG) excels by producing more comprehensive results. This method not only generates a well-reasoned rubric but also assigns an overall grade that aligns closely with the evaluative criteria of the assignment. Although the quality of the generated feedback is comparable to other methods, the ability to elucidate clear grading metrics and detailed rubrics significantly enhances the utility and applicability of the LLM + RG approach in educational settings. This advantage underscores the potential of integrating robust grading frameworks with advanced language models to improve academic assessments.

VI. Conclusion, Limitations and Future Work

A. Conclusion

The outcomes of our experiments indicate that the interactively generated rubrics from the LLM-instructor system can meet a standard comparable to that of manually created rubrics. By automating this task, educators can allocate more

TABLE IV
EXAMPLE OF ONE INTERACTIVELY GENERATED RUBRIC (FROM OPENAI GPT-4).

Term	Points	Description
Description	25	The clarity and the level of detail in the overview, outlining the purpose and functionalities of the code.
Change	20	Degree to which the code has been modified to add new features, fix bugs, or improve efficiency.
Method	20	The implementation and explanation of the method used in the code, including adherence to best practices.
Readable	25	Legibility of the code, includes effective comments, appropriate naming of variables, and overall clean code practices.
Pushed Changes to Github	10	Timeliness and correctness of the updates pushed to Github, includes clear and concise commit messages.

TABLE V
COMPARISON BETWEEN RUBRICS GENERATED BY INTERACTIVELY SELECTING RUBRIC KEYWORDS AND DIRECTLY USING THE KEYWORDS.

Rubric type	The # of keywords	Readability	Clearness	Conciseness
RG + Interactive	5 + 2	9	8	9
RG	5	7	8	8
LLM Only	5	6	7	5

TABLE VI
RUBRIC COMPARISON BETWEEN DIFFERENT LLMS.

Rubric type	# of keywords	Readability	Clearness	Conciseness	Free?
GPT-4	5 + 2	9	8	9	No
Llama-3	5 + 2	8	8	7	Yes
Falcon	5 + 2	6	7	7	Yes
Hermes	5 + 2	6	8	8	Yes

time to direct engagement with students, thereby enriching the learning environment and possibly improving educational outcomes. This finding underscores the potential of LLM technologies in academic settings to streamline and potentially enhance the rubric creation process. These promising results advocate for further exploration and refinement of AI-assisted educational tools.

B. Limitations

While using LDA has provided a foundational method for generating rubric keywords, our experiments have revealed limitations in its effectiveness. Notably, some words extracted by LDA lacked meaningful conceptual relevance, often resulting in a rubric that does not accurately reflect the critical elements of the content. We must filter these irrelevant words out by building a terminology list. Furthermore, the current approach processes all textual materials equally without considering their relative importance. This uniform treatment overlooks the potential benefits of weighting materials differently. For instance, documents directly related to a specific assignment should ideally be prioritized in the weighting process to ensure that the resulting rubric is more pertinent and tailored to the assignment's specific requirements. Another challenge is the scarcity of effective metrics to evaluate the quality of generated data. Due to variations in corpus size and other factors, it is challenging to employ similarity distance metrics, such as Bertscore [27], to compare generated materials with their original counterparts. Additionally, as mentioned by [28], LLMs might have illusion issues, which means the generated text might not be fully correct for all situations. In our work, we let the user participate in the procedure of creating the rubric. Addressing these limitations could lead to more accurate and contextually appropriate rubrics.

TABLE VII
FEEDBACK COMPARISON BETWEEN DIFFERENT METHODS.

Method	Rubric?	Grade?	Readability	Suggestion	Tone
Instructor	No	No	10.0	8.2	9.3
BART from [9]	No	No	9.4	6.6	9.5
LLM(GPT-4)	No	No	9.5	7.0	9.4
LLM + RG	Yes	Yes	9.6	7.1	9.2

C. Future Work

In future work, we intend to implement weighting prioritization within the LDA process to assign importance to various sources of textual materials. This approach may significantly enhance our current strategies for sorting and reclassifying items with priority, which currently relies on the LDA relevance calculation. Additionally, inspired by the work of Guo et al. [29], we intend to implement evolutionary algorithms to optimize our prompt engineering process, potentially improving the efficiency and accuracy of our system. Additionally, as discussed in the previous section, fine-tuning is very helpful in improving the quality of the generated rubric. Another critical area for improvement is the user interface. The existing checkbox interface could be replaced with a more interactive way (for example, a chatbot), to enrich the user experience.

APPENDIX A
PROMPT FOR GENERATING FEEDBACK

The below part shows the prompt pattern we utilize to generate a prompt to send the rubric and the student's report as the input to generate feedback by LLMs.

```
"This is a student peer review environment.
You are an instructor, your task is
to generate instructor feedback from the given
     rubric and a student's report. Do not
    reply until you see the 'END OF REQUEST'".
     If you find one chunk is
missing, please send back which chunk is
    missing
by json style as {"missing":[chunk_number]}."

"The first message is the rubric, you should
    exactly follow the rubric to generate
    feedback to a given report: [Rubric]"

"The following message(s) is (are) the student
    's feedback in chunks:
[Report_Chunk_1/[N]]
[Report_Chunk_2/[N]]
[Report_Chunk_3/[N]]
...
END OF REQUEST"
```

Because some LLMs have a limitation on the number of tokens in one piece of message, we first split the student's report into multiple chunks, then send these chunks respectively to LLM. The LLM will start its task until it receives the phrase "END OF REQUEST" and receives every chunk.

The [Rubric] in the prompt will be automatically substituted with the generated rubric from RG.

REFERENCES

[1] B. Oberer and A. Erkollar, "Mobile learning in higher education: A marketing course design project in austria," *Procedia-Social and Behavioral Sciences*, vol. 93, pp. 2125–2129, 2013, Publisher: Elsevier.

[2] F. Martin, A. Ritzhaupt, S. Kumar, and K. Budhrani, "Award-winning faculty online teaching practices: Course design, assessment and evaluation, and facilitation," *The Internet and Higher Education*, vol. 42, pp. 34–43, Jul. 1, 2019, ISSN: 1096-7516. DOI: 10.1016/j.iheduc.2019.04.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1096751618305669 (visited on 02/18/2024).

[3] M. M.-C. Lu, Y. Hsiao, and H.-H. Wu, "A preliminary study on teacher professional development courses and certification of gender equity education for high school teachers," *Journal of Curriculum Studies*, vol. 17, no. 2, pp. 77–96, 2022, Publisher: Angle Publishing Co., Ltd.

[4] S. J. Yang, H. Ogata, T. Matsui, and N.-S. Chen, "Human-centered artificial intelligence in education: Seeing the invisible through the visible," *Computers and Education: Artificial Intelligence*, vol. 2, p. 100 008, 2021, Publisher: Elsevier.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, May 24, 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs]. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 05/21/2024).

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[7] H. J. Hansen and M. N. Hebart, "Automatic generation of semantic feature norms of objects using GPT-3," *Journal of Vision*, vol. 22, no. 14, p. 3461, Dec. 5, 2022, ISSN: 1534-7362. DOI: 10.1167/jov.22.14.3461. [Online]. Available: https://doi.org/10.1167/jov.22.14.3461 (visited on 05/21/2024).

[8] J. White, Q. Fu, S. Hays, *et al.*, *A prompt pattern catalog to enhance prompt engineering with ChatGPT*, Feb. 21, 2023. DOI: 10.48550/arXiv.2302.11382. arXiv: 2302.11382[cs]. [Online]. Available: http://arxiv.org/abs/2302.11382 (visited on 09/07/2023).

[9] Q. Jia, M. Young, Y. Xiao, *et al.*, "Automated feedback generation for student project reports: A data-driven approach," *Journal of Educational Data Mining*, vol. 14, no. 3, pp. 132–161, Dec. 18, 2022, Number: 3, ISSN: 2157-2100. DOI: 10.5281/zenodo.7304954. [Online]. Available: https://jedm.educationaldatamining.org (visited on 08/01/2023).

[10] S. Moore, R. Tong, A. Singh, *et al.*, "Empowering education with LLMs - the next-gen interface and content generation," in *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, N. Wang, G. Rebolledo-Mendez, V. Dimitrova, N. Matsuda, and O. C. Santos, Eds., Cham: Springer Nature Switzerland, 2023, pp. 32–37, ISBN: 978-3-031-36336-8. DOI: 10.1007/978-3-031-36336-8_4.

[11] OpenAI, *GPT-4 technical report*, Mar. 27, 2023. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774[cs]. [Online]. Available: http://arxiv.org/abs/2303.08774 (visited on 10/16/2023).

[12] M. M. Rahman and Y. Watanobe, "ChatGPT for education and research: Opportunities, threats, and strategies," *Applied Sciences*, vol. 13, no. 9, p. 5783, 2023, Publisher: MDPI.

[13] V. Raina and M. Gales, *Multiple-choice question generation: Towards an automated assessment framework*, Sep. 23, 2022. DOI: 10.48550/arXiv.2209.11830. arXiv: 2209.11830[cs]. [Online]. Available: http://arxiv.org/abs/2209.11830 (visited on 05/20/2024).

[14] Y. Wang, W. Zhong, L. Li, *et al.*, *Aligning large language models with human: A survey*, Jul. 24, 2023. DOI: 10.48550/arXiv.2307.12966. arXiv: 2307.12966[cs]. [Online]. Available: http://arxiv.org/abs/2307.12966 (visited on 05/20/2024).

[15] X. Wu, X. He, T. Liu, N. Liu, and X. Zhai, "Matching exemplar as next sentence prediction (MeNSP): Zero-shot prompt learning for automatic scoring in science education," in *Artificial Intelligence in Education*, N. Wang, G. Rebolledo-Mendez, N. Matsuda, O. C. Santos, and V. Dimitrova, Eds., ser. Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2023, pp. 401–413, ISBN: 978-3-031-36272-9. DOI: 10.1007/978-3-031-36272-9_33.

[16] A. Mizumoto and M. Eguchi, "Exploring the potential of using an AI language model for automated essay scoring," *Research Methods in Applied Linguistics*, vol. 2, no. 2, p. 100 050, Aug. 1, 2023, ISSN: 2772-7661. DOI: 10.1016/j.rmal.2023.100050. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772766123000101 (visited on 05/20/2024).

[17] G. M. Hocky and A. D. White, "Natural language processing models that automate programming will transform chemistry research and teaching," *Digital Discovery*, vol. 1, no. 2, pp. 79–83, 2022, Publisher: Royal Society of Chemistry. DOI: 10.1039/D1DD00009H. [Online]. Available: https://pubs.rsc.org/en/content/articlelanding/2022/dd/d1dd00009h (visited on 05/20/2024).

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, Jan 2003.

[19] C. Sievert and K. Shirley, "LDAvis: A method for visualizing and interpreting topics," in *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, 2014, pp. 63–70.

[20] E. F. Gehringer, "Expertiza: Information management for collaborative learning," *Monitoring and Assessment in Online Collaborative Environments: Emergent Computational Technologies for E-Learning Support*, pp. 143–159, 2009, Publisher: IGI Global Press.

[21] H. Chase, *LangChain*, Oct. 2022. [Online]. Available: https://github.com/langchain-ai/langchain.

[22] H. Touvron, T. Lavril, G. Izacard, *et al.*, *LLaMA: Open and efficient foundation language models*, Feb. 27, 2023. DOI: 10.48550/arXiv.2302.13971. arXiv: 2302.13971[cs]. [Online]. Available: http://arxiv.org/abs/2302.13971 (visited on 05/20/2024).

[23] E. Almazrouei, H. Alobeidli, A. Alshamsi, *et al.*, "Falcon-40b: An open large language model with state-of-the-art performance," 2023.

[24] *Hermes-2-pro-llama-3-8b*, 2023. [Online]. Available: [https://huggingface.co/NousResearch/Hermes-2-Pro-Llama-3-8B] https://huggingface.co/NousResearch/Hermes-2-Pro-Llama-3-8B).

[25] Y. Xiao, G. Zingle, Q. Jia, *et al.*, "Detecting problem statements in peer assessments," in *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[26] G. Zingle, B. Radhakrishnan, Y. Xiao, *et al.*, "Detecting suggestions in peer assessments," in *Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*, International Educational Data Mining Society, Jul. 2019.

[27] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *BERTScore: Evaluating text generation with BERT*, Feb. 24, 2020. DOI: 10.48550/arXiv.1904.09675. arXiv: 1904.09675[cs]. [Online]. Available: http://arxiv.org/abs/1904.09675 (visited on 05/20/2024).

[28] A. Coletta, K. Dwarakanath, P. Liu, S. Vyetrenko, and T. Balch, *LLM-driven imitation of subrational behavior : Illusion or reality?* Feb. 13, 2024. DOI: 10.48550/arXiv.2402.08755. arXiv: 2402.08755[cs,econ,q-fin]. [Online]. Available: http://arxiv.org/abs/2402.08755 (visited on 05/12/2024).

[29] Q. Guo, R. Wang, J. Guo, *et al.*, *Connecting large language models with evolutionary algorithms yields powerful prompt optimizers*, Sep. 15, 2023. DOI: 10.48550/arXiv.2309.08532. arXiv: 2309.08532[cs]. [Online]. Available: http://arxiv.org/abs/2309.08532 (visited on 10/04/2023).