5-2024

Automatic grading of short answers using Large Language Models in software engineering courses

Nguyen Binh Duong TA

Yi Meng CHAI

Automatic Grading of Short Answers Using Large Language Models in Software Engineering Courses

Ta Nguyen Binh Duong, Chai Yi Meng
School of Computing and Information Systems
Singapore Management University
Email: donta@smu.edu.sg

***Abstract*—Short-answer based questions have been used widely due to their effectiveness in assessing whether the desired learning outcomes have been attained by students. However, due to their open-ended nature, many different answers could be considered entirely or partially correct for the same question. In the context of computer science and software engineering courses where the enrolment has been increasing recently, manual grading of short-answer questions is a time-consuming and tedious process for instructors.**

In software engineering courses, assessments concern not just coding but many other aspects of software development such as system analysis, architecture design, software processes and operation methodologies such as Agile and DevOps. However, existing work in automatic grading/scoring of text-based answers in computing courses have been focusing more on coding-oriented questions. In this work, we consider the problem of autograding a broader range of short answers in software engineering courses. We propose an automated grading system incorporating both text embedding and completion approaches based on recently introduced pre-trained large language models (LLMs) such as GPT-3.5/4. We design and implement a web-based system so that students and instructors can easily leverage autograding for learning and teaching. Finally, we conduct an extensive evaluation of our automated grading approaches. We use a popular public dataset in the computing education domain and a new software engineering dataset of our own. The results demonstrate the effectiveness of our approach, and provide useful insights for further research in this area of AI-enabled education.

***Index Terms*—automatic grading, large language models, embedding, software engineering courses, short answers**

I. Introduction

Assessments in education can be done in many forms, for instance multiple-choice questions, essays, short written responses, coding tests, etc. We note that questions which require short textual answers are popular in educational assessments [1]. One of the main reasons is that they could be considered to be more effective compared to multiple-choice questions due to a greater level of information retrieval from memory when trying to come up with answers [2]. However, short-answer questions can accept different correct and partially correct answers. Grading many of such answers is undoubtedly a very tedious and time-consuming process, especially in computing courses at the university level where the number of students has been increasing significantly recently.

Automatic grading/scoring of short textual answers is an established problem in technology-enabled education. Various existing approaches made use of traditional machine learning techniques [3], [4], which require careful feature extractions before model training and score prediction. More recent approaches leverage deep learning techniques, which are able to learn representative features from huge amounts of data instead manual feature engineering work. The deep learning based approaches may suffer from the lack of data on short answer based assessments.

Latest advances in pre-trained LLMs, e.g., OpenAI's release of the GPT family of models, have enabled researchers to further investigate autograding for text-based responses from students, e.g., [5], [6]. However, not much work has been done for LLM-based autograding of short answers in the context of computing education, especially software engineering courses [7]. Such courses cover a wide range of topics including programming, system design, Agile processes, DevOps practices in system deployment, operation, and maintenance, etc. Assessment questions in these topics, e.g., "list one problem with agile processes such as Scrum?" could have a wide range of correct answers. We note that automated grading in computing education has been focusing more on coding based questions [8], [9], which could have a rather limited set of valid responses and could be graded by running pre-determined unit testcases.

In this work, we consider the problem of autograding of short answers in the context of software engineering courses, which are not limited to just programming/coding questions. We makes the following contributions in this paper:

- We propose an automated grading method incorporating both text embedding and completion approaches based on recently introduced pre-trained LLMs such as GPT-3.5-Turbo and GPT-4. The completion-based autograding approach also leverages Retrieval Augmented Generation [10] for better grading accuracy.
- We design and implement a web based system for our LLM-based autograding approaches. The system targets both instructors and students. Instructors can use the web system to do manual adjustments of the autograded scores and to provide additional feedback to answers from students; while students can practice question answering with instant grading.
- We compile a new dataset containing popular questions and short answers in the context of our software engineering courses. These courses cover important software concepts in addition to programming, namely system design,

software testing, Agile processes, DevOps practices, etc. This dataset complements existing ones, e.g., the Mohler dataset [3] which is mainly about programming based questions.

- We conduct an extensive evaluation of our automated grading approach using the new dataset, together with another public dataset in the domain of computer science. To this end, we compare our approach in short-answer grading to some of the most popular existing deep learning based approaches including paragraph embeddings and Siamese long short-term memory (LSTM) neural networks. The results demonstrate the effectiveness of our approach, and provide useful insights for further work in this area.

This paper is organized as follows. Section II discusses related work in short answer autograding, especially recent work in deep learning and LLMs. Section III describes our approaches to autograding of short answers. Section IV provides details on our web based system implementation. Section V presents our evaluation methodology, while Section VI discusses the experimental results. Section VII concludes the paper and highlights possible future work.

II. Related work

Below we summarize several key recent and existing work in automatic short answer grading. We will compare the reported performance for these approaches to ours in this paper where possible.

A. Deep learning based approaches

Traditional machine learning techniques have been applied to the problem of automated short answer grading for many years. In these approaches, e.g., [3], [4], [11], manual feature engineering is needed before training the models on a part of the dataset. For instance, [4] described feature extraction methods including text similarity, question demoting, term weighting, etc. Using these features, a simple ridge regression model was trained. The authors reported autograding performance, e.g., accuracy, in the form of the Pearson correlation coefficient value of 0.592, and the root mean squared error (RMSE) of 0.887. They used a dataset consisting of many computer programing related questions and answers [3] made available by Mohler et al.

Recently, deep learning based approaches have gained much popularity. Deep learning based autograders automatically learn representative features from large datasets. In [12], the authors did a comprehensive survey of deep learning approaches, including embedding, sequential models and attention-based neural networks for short answer grading. The authors then showed that the features learned by deep learning methods mainly work as complementary to manually crafted features of the autograding model. [13] considered automatic grading of short answers using two different types of paragraph embedding models. They obtained a Pearson correlation coefficient of 0.569 and RMSE of 0.797 on the Mohler dataset [3]. Other neural network based approaches were described in [14] and [15], which leverage the Siamese Bidirectional Long Short-Term Memory networks (BiLSTMs). Their results were also reported using the same dataset in [3]. More recent approaches to short answer grading including [16], which uses the Transformer architecture [17] and other optimization techniques to address the problem of insufficient training data.

B. LLM-based approaches

Due to recent advances in pre-trained LLMs, there have been a growing body of work making use of LLMs for automated grading in educational contexts. In particular, [5] investigated text augmentation techniques using GPT-3.5 to improve the dataset for training machine learning models which will be used to provide automated feedback to students. [6] evaluated the accuracy of using GPT-3's *text-davinci-003* model for automatic grading of essays. Using 12,100 essays, it concluded that GPT-3 models, combined with linguistic features, provided a high level of accuracy. Note that this is for essay scoring, not short answer grading in computer science related courses. [18] also used OpenAI's GPT-3.5 *text-davinci-003* model for one-shot prompting and the text completion API to do automatic grading. However, they made use of the Prize Short Answer Scoring dataset, which includes questions from science, biology, English, etc., but not computer science related courses. Similarly, [19] investigated automated scoring for the subject of divergent thinking. The authors performed fine-tuning of LLMs on human-judged responses. The authors of [20] evaluated GPT-4 for short answer grading using the SciEntsBank and Beetle datasets. They found that for these datasets, GPT-4's performance is comparable to manually crafted machine learning models.

Regarding autograding of short answers in the context of computer science related courses, very recent works including [21] which made use of ChatGPT for grading exams in a data science course. They also evaluated ChatGPT for a German-based information system introductory course. They found that such LLM deployment can be valuable, but it is not yet ready for fully automated grading. ChatGPT was also used in [22] to provide corrections to open-ended answers from software development professionals participating in technical training. The authors found that subject matter experts usually agreed with the corrections given by ChatGPT. None of these work made use of well-known datasets in computer science courses such as the Mohler dataset [3]. The exception is [7], in which the authors compared pre-trained LLMs such as ELMo, BERT, GPT-2, etc., directly on their autograding performance for the Mohler dataset. We note that this work was done a while ago so the latest GPT models were not included.

C. Summary

We note that existing deep learning based approaches to short answer grading could provide good accuracy, but they need to be combined with hand-crafted features and require extensive training with large datasets. On the other hand, more recent approaches based on generative AI, in particular pre-trained LLMs, have been focusing more on other educational domains which are not computer science related. In addition, many of the existing approaches made use of the computer science dataset from [3] which had been released a while ago. This dataset is about basic data structures and computer programming concepts.

In this work, we aim to develop new LLM-based approaches which do not require training, and to evaluate these approaches using an entirely new dataset obtained from software engineering courses which include many more topics and concepts beyond just programming. We plan to release our new dataset publicly to encourage further research in this area.

III. LLM-based Auto-Grading Approaches

In this section, we describe in details our proposed approaches to auto-grading short answers, namely the embedding-based, and the completion-based approach. Both of the approaches are based on latest advances in pre-trained LLMs, in particular the text embedding and chat completion models released publicly by OpenAI.

A. Embedding-based

Text embeddings are numerical representations of text in which words or phrases are represented as a vector of numbers. They are used to capture semantic meanings and relationships between words or phrases, enabling more efficient processing and understanding of human languages [23].

```
1  Input: pair of question, answer (Q, A)
2      list R = [reference answers for Q]
3
4  Output: numerical score S for A
5  Steps:
6      Ch = 0
7      Sq = 0
8      Compute the embedding Ea for A
9      For each reference answer Ar in R
10         Compute the embedding Er for Ar
11         Compute a cosine similarity Cr = cos(Er, Ea)
12         If Cr > Ch:
13             Ch = Cr
14             Sq = score of Ar
15
16     S = Ch * Sq
17     Return S
```

Listing 1. Embedding-based autograding approach

The algorithm for our embedding-based autoscoring approach is shown in Listing 1. In this approach, the algorithm computes the embeddings of all the reference answers and student answers for a particular question using an available text embedding model (lines 8-11 of Listing 1). In this work, we use OpenAI's *text-embedding-ada-002* model as it is OpenAI's best and most cost-effective embedding model as of 2023.

The cosine similarity [24] between each reference answer and student answer (to be auto-graded) is then calculated using their corresponding embedding vectors, A and B respectively, as follows:

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \quad (1)$$

The cosine similarity will range from 0 to 1, with 0 being the least similar and 1 being the most similar. After comparing the cosine similarities between each student answer and all the reference answers, the most similar reference answer to the student answer will be selected (lines 12-14 of Listing 1). A mark will then be given to the student answer which is proportional to the cosine similarity (line 16 of Listing 1). This is done by multiplying the cosine similarity score with the reference answer's score.

The embedding-based autoscoring of short answers can be implemented and deployed to use quickly due to the general availability and affordability of of state-of-the-art text embedding models such as *text-embedding-ada-002*. For instance, its pricing as the time of writing is just \$0.0001 per 1K tokens. However, this approach might require a wide range of possible reference answers to be provided for more accurate grading. For short-answer questions, this is potentially challenging as there can be a large number of possibly correct answers to a single question. We can mitigate this issue by using correct answers from students as reference answers. Another issue is that, although models such as *text-embedding-ada-002* is quite affordable, computing embeddings for answers every time you need to do grading (lines 8 and 10 of Listing 1) will add to the total cost. For this, we could use a vector database such as Chroma (https://www.trychroma.com) to store and retrieve the pre-computed embeddings when required.

B. Completion-based

Completion is essentially the generation of output based on the text prompts given to a pre-trained LLM such as GPT-3.5-Turbo. Prompt construction, or prompt engineering for LLMs is an active research area [25]. In a prompt, we may provide relevant instructions, examples, etc., in natural languages. Such data would help direct the model to produce the desired output. One way to do prompting is called zero-shot, in which a query is sent to the LLM without concrete examples of expected results. On the other hand, in few-shot prompting, we provide multiple examples of questions and their corresponding answers in a simulated multi-turn conversation with the LLM. At the end of the conversation, we can ask the LLM to score a student answer for a given question.

In this completion-based autograding approach, we make use of the OpenAI's Chat Completions API[1]. The API defines prompts as sequences of messages. Each message has two components, namely role and content. The role can be "system", "user", or "assistant". A message with "system" role is usually used first to define the behavior of the model. A "user" message gives instructions, and an "assistant" message provides an example of the desired output. The prompt is constructed with all the required messages and sent to the LLM via an API call. Our completion-based autograding approach is shown in Listing 2.

[1]https://platform.openai.com/docs/guides/text-generation/chat-completions-api

```
Message 1: {"role": "system", "content": "You are an
     AI assistant for teaching software engineering
    concepts."}

# Start providing examples in the prompt here
Message 2: {"role": "user", "content": "Given the
    question 'What could be a problem with
    monolithic software?', provide a score for the
    corresponding answer 'Scaling needs to be done
    for the whole application'."}
Message 3: {"role": "assistant", "content": "Score:
    4/4"}

Message 4: {"role": "user", "content": "Given the
    question 'What could be a problem with
    monolithic software?', provide a score for the
    corresponding answer 'It is easier to develop'."
    }
Message 5: {"role": "assistant", "content": "Score:
    1/4"}

# Provide more examples using additional messages if
     needed

# This message is used for autograding
Last message: {"role": "user", "content": "Given the
     question 'What could be a problem with
    monolithic software?', provide a score for the
    corresponding answer 'It is hard to make changes
    .'"}

# The LLM will respond with an appropriate score in
    the below message
Message: {"role": "assistant", "content": "Score: <
    predicted_score>"}
```

Listing 2. Completion-based autograding approach

When instructors need to do autograding, the completion-based approach constructs a sequence of messages as described in Listing 2. Each "user" message provides the question and a corresponding answer, which could be a reference answer, or a student answer. This "user" message is immediately followed by an "assistant" message which has the score given for the answer. Together, this pair of messages provides a concrete example of how scoring should be done for a question and its corresponding answer. For example, in Listing 2, messages 2 and 3 provide a score of 4/4 for the following question/answer pair: "What could be a problem with monolithic software?" / "Scaling needs to be done for the whole application". Similarly, messages 4 and 5 provide another example for the same question with a different answer. We can give more examples to the prompt by providing more of such pair of messages. Finally, the last "user" message in the prompt will provide the answer to be graded for the same question which has been used in the previous examples. Following the Chat Completions API, the LLM, e.g., GPT-3.5-Turbo, will provide a predicted score for this answer.

In the below, we discuss two important considerations for the completion-based autograding approach, namely example selection and the incorporation of RAG (Retrieval Augmented Generation):

Selecting examples for prompt construction: The number of examples in a prompt could be varied. Providing more examples would likely yield better scoring results as the LLM can learn more effectively using the relevant examples. We note that more examples used translates to more cost, as models such as OpenAI's offerings charge based on the number of tokens in the requests and responses. However, in this work we focus on ways to provide more relevant examples to improve grading accuracy rather than cost.

In our completion-based grading approach, we split the answers in a dataset into three different categories, namely low-quality (having low marks), medium-quality (having average to quite decent marks), and high-quality (having full marks). During the automated grading process for a particular question, our algorithm will select a random answer from each answer category and construct the appropriate prompt to be sent to the LLM. The number of answers to be used as examples for each category can be configurable. For instance, in this work we have considered using 1, 2, and 3 answers per category as examples. As a result, the completion-based grading approach can construct prompts having a total of 3, 6, or 9 examples (for 3 categories). We believe that this approach provides the LLM with a better understanding of the grading rubrics for each given question.

Incorporating Retrieval Augmented Generation (RAG): Pre-trained LLMs have been shown to perform well in many common NLP tasks. However, their knowledge base could not be easily revised or expanded beyond simple fine-tuning, and they may hallucinate in their responses [26]. RAG [10], [27] enables a LLM to access external knowledge databases to complete domain-specific tasks with better consistency, reliability and reduced hallucinations. Given an input, e.g., a question, RAG retrieves relevant texts from the specified external knowledge databases, and adds those texts as context to the prompt to be sent to the LLM. With more appropriate context, the LLM can generate output with higher quality.

In the completion-based autograding approach using RAG, we make use of the course content to provide additional context. The aim is to improve grading accuracy and reliability. For our software engineering courses, we make available PDF lecture notes for each topic covered, e.g., automation, software processes, software testing, etc. The lecture notes will then be parsed and partitioned into chunks of texts for which corresponding text embeddings will be computed. Given a specific question to be graded, the most relevant chunks will be retrieved based on comparing the embedding of the question versus the embedding of each chunk. The relevant chunks are then fed into the LLM as the grading context.

IV. Implementation

For instructors and students to take advantage of LLM-based autograding, we design and implement a web system incorporating both the embedding-based and completion-based autograding approaches. The system components are shown in Figure 1.

A. Components

The system is designed for both computing students and instructors at the undergraduate level. The web interface provides functionalities for instructors to create/read/update/delete

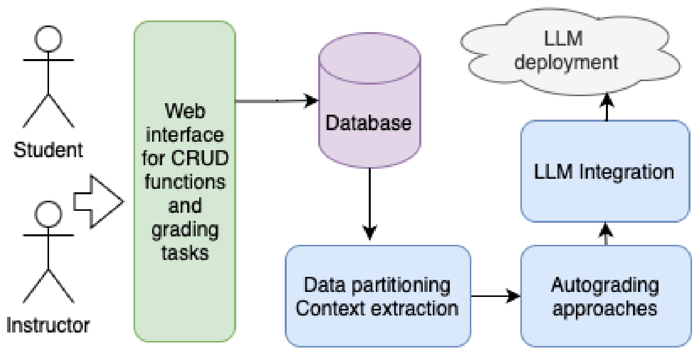

Student
Instructor
Web interface for CRUD functions and grading tasks
Database
LLM deployment
LLM Integration
Data partitioning Context extraction
Autograding approaches

Fig. 1. Components in our web based autograding system

(CRUD) questions and answers, and to monitor student performance. For students, they can use the system as a way to practice for quizzes by answering questions according to topics covered in the course. In this way, students continue to provide more data so the system could get better in autograding over time. The database ensures all the questions/answers/marks are persisted.

We implement a data partitioning mechanism to automatically divide the student answers into categories, e.g., high quality, medium quality, etc., as mentioned in Section III. The mechanism should be rerun once more answers from students have been added to the database. For context extraction, we use OpenAI's *text-embedding-ada-002* and the Faiss library [28], which is a popular package for similarity search developed at Meta's AI Research, to compute and extract chunks of lecture notes relevant to a question which needs to be auto-graded. The context is then incorporated into the prompt together with the grading examples.

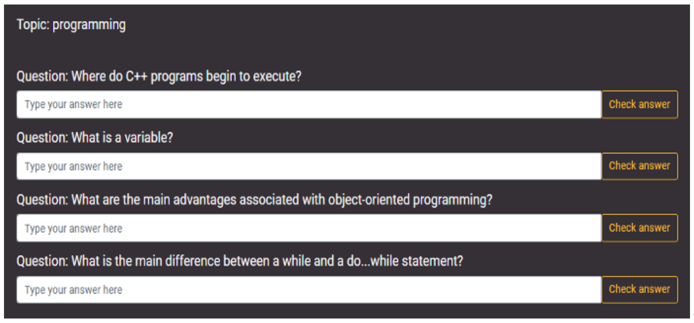
B. LLM deployments

OpenAI's GPT LLMs are used in the implementation of both the embedding-based and completion-based automated grading approaches. These pre-trained LLMs are deployed in the Azure cloud, and accessible via web APIs. In particular, our system accesses the Chat Completions API, and the text embedding via endpoints at *https://api.openai.com/v1/chat/completions*, and *https://api.openai.com/v1/embeddings*, respectively.

In our implementation, we have incorporated three different LLM deployments from OpenAI, namely GPT-3.5-Turbo, GPT-4, and *text-embedding-ada-002*. As these are pay-per-use models, the cost is a concern especially when there are more students using our system in the future. For this reason, GPT-3.5-Turbo is used as the default LLM most of the time instead of GPT-4, as the former is quite capable and cost-effective, i.e., 30x cheaper compared to the latter. The embedding model provided by OpenAI is rather inexpensive, costing just $0.0001 per 1K tokens.

C. Web-based implementation

We have implemented a complete web based system using Vue.js for the frontend interface, Flask for the backend logic, and MongoDB as the database. Figure 2 shows the web interface in which students can practice answering short questions. When students answer a question, their answers and the marks given by our autograding approaches will be automatically added into the database. In Figure 3, the instructors can edit any answers and marks given by the autograding approaches, as well as providing additional feedback for each answer. Instructors can also add more questions/answers for students to practice.



Topic: programming

Question: Where do C++ programs begin to execute?

Type your answer here | Check answer

Question: What is a variable?

Type your answer here | Check answer

Question: What are the main advantages associated with object-oriented programming?

Type your answer here | Check answer

Question: What is the main difference between a while and a do...while statement?

Type your answer here | Check answer

Fig. 2. Students can practice on short answer questions. Their answers will be graded automatically.

V. Evaluation Methodology

This section describes the datasets and the performance measures used in our evaluation.

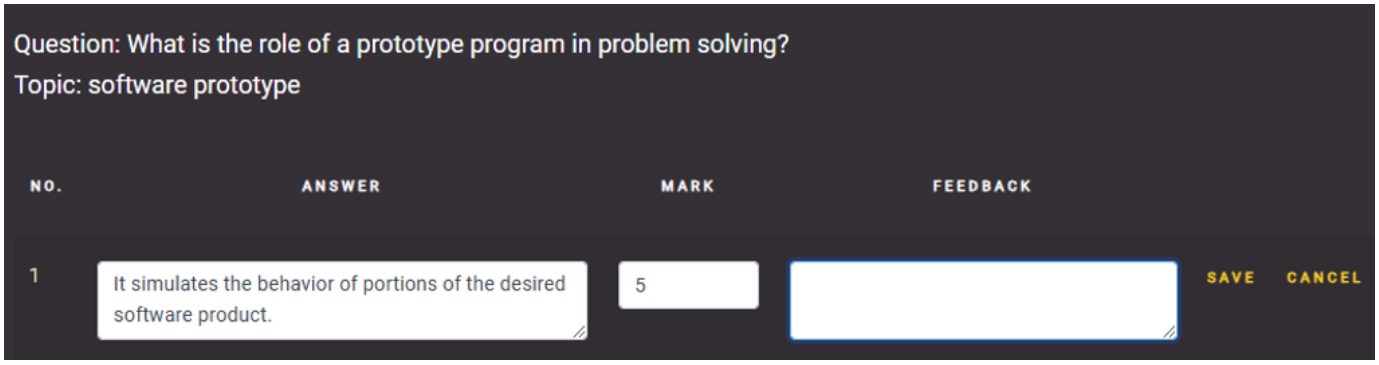
A. Datasets

Two complementary datasets are used to evaluate the performance of our proposed embedding-based and completion-based autograding approaches.

Mohler dataset [3]: This dataset has been widely used in evaluating automatic grading approaches for short answers. Most questions in the dataset are about programming/coding concepts. We use it mainly for fair comparisons with existing approaches in this area. The dataset is obtained through exams/assignments given to students in an introductory computer science class at the University of North Texas. For every student answers, it is marked by two graders and the average mark is calculated for each answer in the range of 0 mark to 5 marks, with 5 marks being the maximum.

The dataset consists of a total of 87 questions and 1 reference answer for each question, but 6 questions are excluded from the dataset as they are not short answer questions. There are 24 to 31 students' answers per questions in the dataset, summing up to 2273 answers with an average of 28 answers per questions. All results obtained through this dataset are based on the 81 questions and 2273 answers. A sample question and its corresponding answers/scores extracted from [3] are shown in Table I.

In our work, the answers in this dataset are split into 3 different categories: low-quality (less than or equal to 2 marks), medium-quality (less than or equal to 4 marks) and high-quality (5 marks). This partitioning is important for the


Question: What is the role of a prototype program in problem solving?
Topic: software prototype

NO.	ANSWER	MARK	FEEDBACK
1	It simulates the behavior of portions of the desired software product.	5	

Fig. 3. Instructors can edit answers and marks given automatically for any question, as well as providing more feedback.

TABLE I
MOHLER DATASET: SAMPLE QUESTION, ANSWERS AND SCORES

Question	What is a pointer?
Reference Answer	A variable that contains the address in memory of another variable.
Student Answer	a pointer holds a memory location.
Score 1	5
Score 2	4
Average Score	4.5

evaluation of the completion-based approach, where different numbers of examples are used for prompting LLMs.

Software engineering (SE) dataset: This is a dataset on the broader topic of software development with subtopics consisting of automation, software design, versioning, agile processes, extreme programming (XP), security, solution support, and testing. The summary for each subtopic is listed in Table II. It nicely complements the Mohler dataset, which is mainly about programming. The dataset consists of a total of 32 short-answer questions, with the number of reference answers per question ranging from 1 to 4. There is a total of 421 graded answers with their corresponding marks, with an average of 13 answers per question. The marks for each question ranges from 0 to 4, with 4 marks being the maximum. Along with this dataset, there are PDF lecture notes for each of the subtopics with the number of pages ranging from 22 to 50. The PDFs are to be used as additional contexts for the grading of questions related to their respective subtopics.

The answers in this dataset are also split into 3 different mark categories: low-quality (less than or equal to 1 marks), medium-quality (less than or equal to 3.5 marks) and high-quality (4 marks). An example is shown in Table III.

B. Performance measures

Similar to existing work in this area [15], the results are evaluated using the Pearson correlation coefficient, mean absolute error (MAE) and root mean square error (RMSE).

The **Pearson correlation coefficient** is one of the most common way to measure linear correlations. The result will range from value of -1 to 1 depending on the strength and direction of the relationship. A larger absolute value will signify stronger correlation between the two variables tested.

TABLE II
SE LECTURE NOTES INCORPORATED AS GRADING CONTEXTS

Topic	Summary	Pages
Automation	Software deployment models, infrastructure and CI/CD	25
Software design	Dependency injection, REST API design	32
Software processes	Waterfall, iterative and agile processes	29
Security	Confidentiality, integrity, availability approaches	30
Versioning	Distributed version control, Git workflows	36
XP practices	Code review, refactoring, and pair programming	30
Software support	Events, incidents and problem management for software systems	50
Software testing	Blackbox, whitebox, input space partitioning, unit, integration, regression testing	22

TABLE III
SE DATASET: SAMPLE QUESTION, ANSWERS AND SCORE

Subtopic	Automation
Question	What is one advantage of canary deployment?
Reference Answer	Can minimize the impact of errors to a subset of users
Graded Answer	it is cheaper to do
Graded Answer's Score	1

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{2}$$

where x_i represents the actual mark given by human graders, and y_i represent the mark given by the autograding approach for the same answer. $\overline{x}$ and $\overline{y}$ represent the means of x and y, respectively.

The **mean absolute error** (MAE) is calculated by averaging the absolute differences between the actual and predicted marks:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \tag{3}$$

Finally, the **root mean square error** (RMSE) is also used widely to measure the quality of predictions:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \quad (4)$$

In (3) and (4), n is the total number of answers being evaluated, x_i is the actual mark for the i-th answer and y_i is the predicted mark given by the autograding approach for the same answer. Both MAE and RMSE are reliable metrics for assessing the accuracy of predictions.

C. Research questions

In the evaluation, we aim to answer the following research questions (RQs):

- **RQ1**: Which is the better approach for autograding short answers: embedding-based or completion-based?
- **RQ2**: How do embedding-based and completion-based autograding compare to existing deep learning based approaches?
- **RQ3**: Does adding context from relevant lecture notes on the question's topic using RAG produce more accurate grading result when using the completion-based approach?
- **RQ4**: How do different versions of the same LLM family, e.g., GPT-3.5-Turbo and GPT-4, compare to each other in the autograding of short answers?

VI. Results

A. RQ1: Embedding-based vs. completion-based

We first compare our proposed embedding-based and completion-based approaches using both the Mohler dataset [3] and the SE dataset. The default LLM used is GPT-3.5-Turbo. The results are shown in Table IV.

TABLE IV
EMBEDDING VS. COMPLETION

Model	Pearson Correlation Coefficient	RMSE	MAE
Mohler Dataset			
Embedding	0.557	0.932	0.749
Completion (3 examples)	0.450	1.185	0.960
Completion (6 examples)	0.406	0.975	0.780
Completion (9 examples)	0.525	0.922	0.706
SE Dataset			
Embedding	0.507	2.017	1.727
Completion (3 examples)	0.621	1.342	1.044
Completion (6 examples)	0.694	1.207	0.872
Completion (9 examples)	0.674	1.240	0.852

For the Mohler dataset [3], the embedding-based approach produces the highest Pearson correlation coefficient of 0.557. On the other hand, the completion-based approach with 9 examples produces a Pearson correlation of 0.525, as well as the best RMSE and MAE of 0.922 and 0.706 respectively.

For the SE dataset, the results produced by the completion-based approach with 6 examples are quite similar to those produced by the same approach with 9 examples. The former having a higher Pearson correlation coefficient of 0.694 and a lower RMSE of 1.207; while the latter having a lower MAE of 0.852. However, given that more examples have to be passed into the prompt, it may take a longer processing time and higher cost when doing grading. Therefore, in this case the completion-based approach with 6 examples will potentially be more useful due to its balance between efficiency and accuracy.

We can observe a large discrepancy between the results produced by the embedding-based approach for the two different datasets, where the RMSE and MAE for the SE dataset are more than twice of those produced for the Mohler dataset. In particular, for the Mohler dataset, the embedding-based approach produced a RMSE and a MAE of 0.932 and 0.749, respectively. On the other hand, for the SE dataset, the same approach produced a RSME and a MAE of 2.017 and 1.727. For the Pearson correlation coefficient, it's 0.557 and 0.507 for the Mohler and SE datasets, respectively.

From the experiments, we note that the embedding-based approach is more biased towards giving higher scores due to the relatively high cosine similarities obtained between the student answers and the reference answers in many cases; unless the student's answer is hardly related to the question text. This can be observed from the fact that 86% of the scores predicted for the Mohler dataset [3] is more than 4 marks when using the embedding-based approach. At the same time, we also note that the Mohler dataset has about 63% of the answers scoring above 4 out of 5 marks as given by the human graders. On the other hand, the SE dataset only has 27% of the answers scoring above 3 out of 4 marks. This explains why the embedding-based approach does better in the Mohler dataset, but much worse in the SE dataset. In this case, we observe that the completion-based approach is the better way to do autograding of short answers as it significantly outperforms the embedding-based approach in the SE dataset.

Summary-RQ1: The completion-based approach could be considered the better autograding approach overall; as it is more consistent with the predicted marks given to answers in both datasets, regardless of the actual mark distribution in any of them. In both cases, we will need to provide more relevant examples of answers and actual scores in the completion-based prompt to improve the autograding performance.

B. RQ2: Comparison to deep learning based methods

We now compare the embedding and completion based approaches with other existing autograding methods which made use of deep learning techniques. For a fair comparison, all the approaches are considered using performance measures reported previously with the Mohler dataset. Table V summarizes the key results. In particular, we collected results reported from [14], which implemented and evaluated several variations of the Long Short-Term Memory (LSTM) neural networks for short answer grading. The authors of [13] considered different types of paragraph embedding models. Finally, the usage Bidirectional LSTM (BiLSTM) has been shown to perform well in automated grading [15]. More recently, pre-trained models such as ELMo [7] was also used on the Mohler dataset.

As observed from Table V, the embedding and completion based approaches may not be the best approach in terms of

TABLE V
COMPARING WITH EXISTING DEEP LEARNING BASED AUTOGRADING APPROACHES - MOHLER DATASET

Approach	**Pearson Correlation Coefficient**	**RMSE**	**MAE**
LSTM-EMD-SVOR [14]	0.550	0.830	0.490
LSTM-EMD-Logits [14]	0.649	1.135	0.657
Paragraph embedding (doc2vec) [13]	0.569	0.797	-
Siamese BiLSTM + feature engineering [15]	0.655	0.889	0.618
Stacked BiLSTM (ELMo) [7]	0.485	0.978	-
Embedding-based	0.557	0.932	0.749
Completion-based (9 examples)	0.525	0.922	0.706

result. However they offer a good balance among all the three metrics, and can be seen to have an average performance which is quite comparable to the existing deep learning based approaches, e.g., [13], [14]. It is important to note that in our approaches, there was no extensive training or fine-tuning done with a large labelled dataset. On the other hand, existing deep learning based approaches incur significant training cost with a large part of the same dataset prior to predictions [16]. In some cases, e.g., [15], manual feature engineering tasks are needed to improve the prediction performance.

We note that popular pre-trained LLMs such as BERT and ELMo [29] have been applied on the Mohler dataset. As shown in Table V, the performance of the ELMo-based approach still has a gap when compared to the embedding and completion based approach. There have been research on how to leverage the GPT family of models on short answer grading, e.g., [20], [21]. However, we could not find other recent works making use of the latest pre-trained LLMs like GPT-3.5-Turbo or GPT-4 on the Mohler dataset.

Summary-RQ2: The embedding and completion based approaches do not require extensive training or fine-tuning to perform reasonably well. Therefore, they are more generally applicable to a wide variety of grading scenarios; not just in our specific SE courses but also in other courses.

C. RQ3: Using course materials as additional context in completion-based autograding

We would like to find out if there will be an improvement in the autograding capability of the completion-based approach when provided with more context extracted from relevant course materials such as lecture notes when they are available. This is referred to as RAG - retrieval augmented generation [27]. As described in the implementation, we use OpenAI's *text-embedding-ada-002* and the Faiss library [28] to store and extract chunks of lecture notes (as detailed in Table II) relevant to a question which needs to be auto-graded. The context is then incorporated into the prompt together with the grading examples.

TABLE VI
EVALUATING THE EFFECT OF ADDITIONAL CONTEXT IN AUTOGRADING - SE DATASET

	Pearson Correlation Coefficient	**RMSE**	**MAE**
Completion (3 examples)	0.621	1.342	1.044
Completion with context (3 example)	0.631	1.338	1.018
Completion (6 examples)	0.694	1.207	0.872
Completion with context (6 examples)	0.642	1.149	0.795
Completion (9 examples)	0.674	1.240	0.852
Completion with context (9 examples)	0.748	1.026	0.693

The results are shown in Table VI for the SE dataset, for which we have the corresponding course materials. With relevant context given in the prompt, it is observed that there are generally notable improvements in the quality of autograding for short answers. For instance, the completion-based approach with 9 examples (no context provided) produces a Pearson correlation coefficient of 0.674, RMSE of 1.207 and MAE of 0.872. With the context, the same approach produces better predictions with a Pearson correlation coefficient of 0.748, RMSE of 1.026 and MAE of 0.693. The only exception is the completion-based (6 examples), in which the additional context does not improve the already high Pearson correlation. However, Table VI shows that the RMSE and MAE have been improved due to more context in that case.

Summary-RQ3: Relevant context extracted from course materials and given to the LLM prompt could significantly improve the autograding accuracy in most cases.

D. RQ4: Comparison between GPT-3.5-Turbo and GPT-4

We also compare the grading performance of the GPT-4 and GPT-3.5-Turbo LLMs. We note that the cost of GPT-4 is significantly more than GPT-3.5-Turbo for the same amount of tokens. Due to the limited budget, we have not had the chance to fully explore GPT-4's capabilities. In our system, we use GPT-4 to implement the completion-based approach with 6 and 9 examples, and evaluate it on the SE dataset.

In Table VII, there is a significant improvement in the GPT-4 based approach when compared to the one using GPT-3.5-Turbo. GPT-4 completion-based approach with 9 examples achieved the highest Pearson correlation coefficient of 0.844, a low RMSE and MAE of 0.828 and 0.566, respectively. While the results are very promising, a more extensive evaluation of GPT-4 based approaches is needed when the cost becomes less of an issue. It is worth noting that as the time of writing, GPT-4 generally costs about 30x more compared to GPT-3.5-Turbo.

TABLE VII
GPT-4 vs. GPT-3.5-Turbo for the completion-based approach - SE dataset

	Pearson Correlation Coefficient	RMSE	MAE
GPT-3.5-Turbo (6 examples)	0.694	1.207	0.872
GPT-4 (6 example)	0.784	0.896	0.616
GPT-3.5-Turbo (9 examples)	0.674	1.240	0.852
GPT-4 (9 examples)	0.844	0.828	0.566

Summary-RQ4: The newer LLM version such as GPT-4 could significantly outperform previous models in short answer autograding.

E. Limitations

Here we discuss some limitations of this work. First, outputs from LLMs could vary from time to time, which might affect the autograding accuracy reported in Section VI. We have attempted to mitigate this issue by reporting the accuracy using a large number of answers from two different datasets. Second, due to funding issues we could not fully evaluate GPT-4's autograding accuracy. It is possible that newer and more expensive LLM versions will provide improved performance compared to what we have reported here. Finally, it would be better to build a larger dataset with more questions and answers on various software engineering topics. We plan to do so with the latest LLM versions, e.g., GPT-4 Turbo, when the cost becomes more manageable.

VII. Conclusion

This work on LLM-based automatic grading of short answers has the potential to reduce the marking burden on instructors teaching a variety of courses, especially in the domain of computer science and software engineering where the number of students has been increasing recently. We have proposed two new approaches for autograding short answers using embedding and completion models, which are based on the OpenAI's GPT family of LLMs.

We have conducted extensive evaluations and comparison to the existing methods in this area using a well-known dataset and a new dataset of our own in software engineering courses at the university level. The datasets capture different kinds of mark distributions which could affect any autograding methods. We found that our approaches, especially the completion-based approach, which do not require time-consuming training of deep learning models, could work well for the given datasets. We also found that relevant context in the form of lecture notes for the course would help improve grading performance. Lastly, newer models like GPT-4 look very promising in autograding tasks. However, the cost of such models is still a concern especially for educational institutions. We plan to investigate ways to do more accurate and fair autograding while minimizing LLM cost in our future work.

Acknowledgement

This work is supported by the UResearch programme from the School of Computing and Information Systems, Singapore Management University.

References

[1] T. Puthiaparampil and M. M. Rahman, "Very short answer questions: a viable alternative to multiple choice questions," *BMC medical education*, vol. 20, no. 1, pp. 1–8, 2020.

[2] S. Greving and T. Richter, "Examining the testing effect in university teaching: Retrievability and question format matter," *Frontiers in Psychology*, vol. 9, 2018.

[3] M. Mohler, R. Bunescu, and R. Mihalcea, "Learning to grade short answer questions using semantic similarity measures and dependency graph alignments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 752–762.

[4] M. A. Sultan, C. Salazar, and T. Sumner, "Fast and easy short answer grading with high accuracy," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1070–1075.

[5] K. Cochran, C. Cohn, J. F. Rouet, and P. Hastings, "Improving automated evaluation of student text responses using gpt-3.5 for text data augmentation," in *International Conference on Artificial Intelligence in Education*. Springer, 2023, pp. 217–228.

[6] 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, 2023.

[7] S. K. Gaddipati, D. Nair, and P. G. Plöger, "Comparative evaluation of pretrained transfer learning models on automatic short answer grading," *arXiv preprint arXiv:2009.01303*, 2020.

[8] J. Mitra, "Studying the impact of auto-graders giving immediate feedback in programming assignments," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 388–394.

[9] D. S. Mishra and S. H. Edwards, "The programming exercise markup language: Towards reducing the effort needed to use automated grading tools," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 395–401.

[10] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, 2023.

[11] S. Basu, C. Jacobs, and L. Vanderwende, "Powergrading: a clustering approach to amplify human effort for short answer grading," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 391–402, 2013.

[12] S. Haller, A. Aldea, C. Seifert, and N. Strisciuglio, "Survey on automated short answer grading with deep learning: from word embeddings to transformers," *arXiv preprint arXiv:2204.03503*, 2022.

[13] S. Hassan, A. A. Fahmy, and M. El-Ramly, "Automatic short answer scoring based on paragraph embeddings," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, 2018.

[14] S. Kumar, S. Chakrabarti, and S. Roy, "Earth mover's distance pooling over siamese lstms for automatic short answer grading," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 2046–2052.

[15] A. Prabhudesai and T. N. Duong, "Automatic short answer grading using siamese bidirectional lstm based regression," in *2019 IEEE international conference on engineering, technology and education (TALE)*. IEEE, 2019, pp. 1–6.

[16] X. Zhu, H. Wu, and L. Zhang, "Automatic short-answer grading via bert-based deep neural networks," *IEEE Transactions on Learning Technologies*, vol. 15, no. 3, pp. 364–375, 2022.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] S.-Y. Yoon, "Short answer grading using one-shot prompting and text similarity scoring model," *arXiv preprint arXiv:2305.18638*, 2023.

[19] P. Organisciak, S. Acar, D. Dumas, and K. Berthiaume, “Beyond semantic distance: automated scoring of divergent thinking greatly improves with large language models,” *Thinking Skills and Creativity*, p. 101356, 2023.
[20] G. Kortemeyer, “Performance of the pre-trained large language model gpt-4 on automated short answer grading,” *arXiv preprint arXiv:2309.09338*, 2023.
[21] J. Schneider, B. Schenk, C. Niklaus, and M. Vlachos, “Towards llm-based autograding for short textual answers,” *arXiv preprint arXiv:2309.11508*, 2023.
[22] G. Pinto, I. Cardoso-Pereira, D. Monteiro, D. Lucena, A. Souza, and 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.
[23] J. M. Gomez-Perez, R. Denaux, A. Garcia-Silva, J. M. Gomez-Perez, R. Denaux, and A. Garcia-Silva, “Understanding word embeddings and language models,” *A Practical Guide to Hybrid Natural Language Processing: Combining Neural Models and Knowledge Graphs for NLP*, pp. 17–31, 2020.
[24] P. Xia, L. Zhang, and F. Li, “Learning similarity with cosine similarity ensemble,” *Information sciences*, vol. 307, pp. 39–52, 2015.
[25] P. Denny, V. Kumar, and N. Giacaman, “Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language,” in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 1136–1142.
[26] A. Martino, M. Iannelli, and C. Truong, “Knowledge injection to counter large language model (llm) hallucination,” in *European Semantic Web Conference*. Springer, 2023, pp. 182–185.
[27] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
[28] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.
[29] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, “Recent advances in natural language processing via large pre-trained language models: A survey,” *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.