Insta-Reviewer: A Data-Driven Approach for Generating Instant Feedback on Students' Project Reports

Qinjin Jia, Mitchell Young, Yunkai Xiao, Jialin Cui, Chengyuan Liu, Parvez Rashid, Edward Gehringer
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
qjia3, mgyoung, yxiao28, jcui9, cliu32, mrashid4, efg@ncsu.edu

ABSTRACT

Providing timely feedback is crucial in promoting academic achievement and student success. However, for multifarious reasons (e.g., limited teaching resources), feedback often arrives too late for learners to act on the feedback and improve learning. Thus, automated feedback systems have emerged to tackle educational tasks in various domains, including novice programming, short-essay writing, and open-ended questions. However, to the best of our knowledge, no previous study has investigated automated feedback generation on students' project reports. In this paper, we present a novel data-driven system, named *Insta-Reviewer*, for automatically generating instant feedback on students' project reports, using state-of-the-art natural language processing (NLP) models. We also propose a framework for manually evaluating system-generated feedback. Experimental results show that feedback generated by Insta-Reviewer on real students' project reports can achieve near-human performance. Our work demonstrates the feasibility of automatic feedback generation for students' project reports while highlighting several prominent challenges for future research.

Keywords

Feedback generation, automated review generation, instant feedback, learning at scale, mining educational data

1. INTRODUCTION

Feedback plays a vital role in the student learning process, as it can help students reinforce or correct their understanding of knowledge and content by giving them clear guidance on how to improve their learning [1, 19, 22, 39]. Furthermore, instant feedback is usually more effective than delayed feedback, presumably because timely feedback is more likely to motivate students to stay on task and encourage them to achieve learning goals [21, 51, 14]. However, owing to various constraints (e.g., staff availability), feedback often comes too late for students to enact the advice and benefit their learning [48, 38, 32, 22]. Students reported in a prior study that delayed feedback is perceived as irrelevant because it has been so long that they have forgotten about the content, which discourages them from following the feedback [38]. Thus, tardy feedback can unintentionally position students as passive recipients of feedback information and limit their engagement with feedback and learning [8, 38].

One way of bringing about the much-needed immediacy is by way of automatic generation of instant feedback on students' work. Thanks to recent technological advancements, a variety of automatic feedback systems have emerged to tackle educational tasks in various domains, including novice programming [31, 56], short-essay writing [49, 5], and open-ended short answers [28, 3]. For instance, Malik and Wu proposed generative models for providing feedback on short answers and different types of programming assignments [28]. Marwan et al. designed a hybrid method to deliver instant feedback for block-based programming [31]. In addition to this, many other impressive studies focus on educational tasks that demand instant feedback to facilitate students' learning and show promising results across modalities and domains (e.g., [27, 36, 47, 2, 29]). It can be argued that automatic feedback systems will be integral parts of the future AI-powered educational ecosystem [41].

However, to the best of our knowledge, no attempt has been made to evaluate the feasibility of automatic feedback generation on students' project reports. It is well-known that course projects are an essential part of many university curricula, especially STEM courses [4, 17]. These projects can help students reinforce their theoretical knowledge and develop a host of skills that are increasingly important in the professional world [30, 17]. However, delivering immediate feedback on project reports is often infeasible for instructors. We summarize the reasons why such an automated feedback system for students' project reports is significant as follows:

1. Despite the fact that course projects have many positive educational impacts on students, the burden of providing timely feedback may prevent instructors from offering sufficient course projects. In this case, an automated feedback system can encourage instructors to provide more project work in classes.
2. Many instructors can merely provide summative feedback for a final project at the end of a semester, which does not give students an opportunity to implement the advice. However, if an automated feedback system is available to provide formative feedback, students will

Q. Jia, M. Young, Y. Xiao, J. Cui, C. Liu, P. Rashid, and E. Gehringer. Insta-reviewer: A data-driven approach for generating instant feedback on students' project reports. In A. Mitrovic and N. Bosch, editors, *Proceedings of the 15th International Conference on Educational Data Mining*, pages 5–16, Durham, United Kingdom, July 2022. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.6853099

have guidance on how to revise their work and reinforce their learning without adding workload to instructors.

3. An automated feedback system can help promote educational equity and diversity by giving students the benefit of quality feedback on projects in an institution that has high teaching loads and limited or nonexistent teaching assistant support, or even in a MOOC.

In this paper, we present a data-driven system, named *Insta-Reviewer*, for generating instant textual feedback on students' project reports. Insta-Reviewer utilizes a select-then-generate paradigm consisting of two main steps: 1) the paradigm first uses an unsupervised method, called cross-entropy extraction, to summarize original reports to lengths acceptable for input into our text generation model used in the second step, and then 2) employs a supervised text-to-text generation model called BART to generate plausible and readable textual feedback for the corresponding report. In order to explore the quality of generated feedback, we employ a comprehensive set of evaluation metrics, including a content-overlap metric ROUGE, a model-based metric BERTScore, and a new human-centered evaluation metric.

To investigate the potential promise of our system, we design experiments to answer the following research questions:
RQ1: How effective is the proposed approach for generating feedback on students' project reports?
RQ2: What are the problems of system-generated reviews? What are they not good at?
RQ3: How does the system perform compared with other potential methods?
RQ4: How does the automated feedback system perform in different small-data settings?
RQ5: Does the Insta-Reviewer automated feedback system raise any ethical concerns?

Our results show that feedback generated by Insta-Reviewer on real students' project reports can achieve near-human performance, while it may include some non-factual or ambiguous statements in generated feedback. Our work demonstrates the feasibility of automated instant-feedback generation on students' project reports. Experimental results also highlight several major challenges for future research.

Our main contributions are: 1) we present an effective data-driven approach for generating feedback on students' project reports; 2) we collect a new dataset of students' reports and expert reviews to facilitate future research endeavors; 3) we propose a new framework for manually evaluating generated feedback; 4) we evaluate the effectiveness of our approach in different small-data settings to help others who intend to apply the approach to their datasets; 5) we highlight several prominent challenges for future research.

The remainder of the paper is organized as follows: Section 2 presents related work. Section 3 describes the dataset used for this study. Section 4 elaborates our methodology for automatically generating feedback for students' project reports and explains the new human-evaluation metric. Section 5 presents and discusses our experimental results. Section 6 concludes the paper, mentions the limitations of our work and provides some discussion about future research.

2. LITERATURE REVIEW

In the following, we first review prior studies on automated feedback systems. Then we survey potential metrics for evaluating system-generated feedback. Finally, we review previous work related to ethical concerns in feedback generation.

2.1 Automated Feedback Systems

In the field of education, feedback is defined as information provided by an agent (e.g., teacher, peer) about a learner's performance or understanding, and it is one of the most significant influences on student learning and achievement [19]. Previous research has been devoted to designing a variety of automated feedback systems that provide feedback on various forms of student work, such as short-answer questions, essays, and programming problems. Although these efforts were not intended to provide feedback on student project reports, we reviewed these studies to gain some insight. The feedback-generation models (i.e., the feedback engines) used in previous studies on automated feedback systems for student work can be broadly categorized into two groups:

Expert-driven methods: Expert-driven methods (also called rule-based methods) use a set of expert-designed rules to provide feedback. For instance, Narciss et al. [34] presented an intelligent tutoring system for students learning mathematics. The system uses a set of pre-defined rules provided by domain experts to deliver feedback for students' answers to numerical or multiple-choice questions. Nagata et al. [33] introduced an approach for leveraging the expert-driven method to diagnose preposition errors and produce feedback for learners' English writing. While expert-driven methods are typically accurate and not data-hungry, they are not suitable for dealing with complex open-ended problems (e.g., generating feedback for students' project reports) because creating and maintaining a vast set of expert-design rules for such open-ended problems is nearly impossible. Additionally, these methods usually produce feedback that is limited to fixed expressions without dynamic explanations.

Data-driven methods: Recent technological advances in artificial intelligence have enabled the development of various data-driven automated feedback systems to produce feedback for more complex open-ended tasks. Data-driven methods generate feedback by learning the mappings (i.e., patterns) from student work to expert feedback by means of machine-learning or deep-learning algorithms [10]. For example, Lu and Cutumisu [27] implemented several deep-learning models, including CNN, CNN+LSTM, and CNN+Bi-LSTM, for generating textual feedback on students' essays. However, traditional deep-learning models usually fail to capture long-span dependencies in long documents and rely on large amounts of training data. More recent work has begun to use large-scale pre-trained language models, such as BERT [11], BART [23], and GPT-2 [40], for generating feedback on open-ended student work. These language models use the attention mechanism [45] to learn long-span dependencies and are pre-trained on large generic corpora in an unsupervised manner to reduce the need for labeled data. For instance, Olney [35] attempted to generate elaborated feedback for student responses using the ELI5 model and achieved promising results. In this paper, we also use such a pre-trained language model to exploit its ability to capture long-span dependencies in student reports.

2.2 Evaluation of Feedback Generation

Effective metrics for evaluating generated feedback are essential since we use them to compare different approaches and quantify the progress made on this research problem. However, unlike other tasks (e.g., text classification), accurately evaluating system-generated feedback (and many other natural language generation problems) is in itself a huge challenge, mainly because generating feedback is a highly open-ended task. For instance, an automated feedback system can generate multiple plausible reviews for the same student report, but all these reviews can be vastly different.

All existing evaluation methods for natural language generation tasks can be grouped into three categories: 1) content-overlap metrics, 2) model-based metrics, and 3) human-centered evaluation metrics. Content-overlap metrics and model-based metrics automatically evaluate a text-generation system by measuring the similarity between generated texts and reference texts provided by domain experts. Human-centered evaluation asks people to assess the quality of system-generated texts against set task-specific criteria [9].

It is worth noting that the ultimate goal of our Insta-Reviewer automated feedback system is to generate feedback that is valuable to students instead of generating the exact same feedback as provided by instructors. For this reason, human-authored evaluation should be viewed as the gold standard when evaluating generated feedback. However, human evaluations are inconsistent and subjective, which can lead to erroneous conclusions or prevent researchers from comparing results across systems [9]. Thus, we also employ a content-overlap metric and a model-based metric to validate our human-evaluation results. In the following paragraphs, we survey potential metrics that can be applied to our task.

Content-overlap metrics: Content-overlap metrics refer to a set of metrics that evaluate the quality of generation by comparing generated texts with reference texts (e.g., ground-truth feedback provided by instructors) based on the content overlap, such as n-gram match. Despite the fact that this set of metrics has many limitations (e.g., they do not take synonyms into account), text-generation research usually uses metrics from this set for benchmark analysis with models since they are objective and fast to calculate. Two of the most commonly used content-overlap metrics for evaluating generated text are BLEU [37] and ROUGE [25].

BLEU (the Bilingual Evaluation Understudy) is a precision-based content-overlap metric proposed by Papineni et al. [37] in 2002. The precision-based metric means that we compare the two texts by counting the number of words or n-grams in the generated text that occur in the human reference and dividing the count by the length of the reference text. Compared to ROUGE, BLEU focuses on precision and is suitable for tasks that favor high precision. However, recall is also essential for our task since we expect more words from the expert feedback to appear in the system-generated feedback. On the other hand, both recall and precision can be taken into account for the ROUGE metric [7].

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the other most commonly used content-overlap metric introduced by Lin [25] in 2004. The original ROUGE score is basically a recall-based version of BLEU. That is, for ROUGE, we check how many words or n-grams in the reference text appear in the generated text. Nevertheless, entirely removing precision can have substantial adverse effects (e.g., a system may generate extremely long text strings to capture all words in the reference text). In recent years, ROUGE has commonly referred to ROUGE-F_1 that combines both precision and recall ROUGE scores in the harmonic mean. We report ROUGE-F_1 scores in results since both recall and precision are vital for our system.

Model-based metrics: Model-based metrics use learned representations of words and sentences to compute semantic similarity between generated and reference texts. Model-based metrics are generally more correlated with human judgment than content-overlap metrics, but their behavior is not interpretable [9]. Given the excellent performance of BERT [11] across many tasks, recent work on model-based metrics commonly uses pre-trained contextual embeddings from BERT or similar pre-trained language models for evaluating the semantic equivalence between the texts [55, 53, 43]. One of the most widely used metrics is BERTScore [55].

BERTScore is a model-based metric proposed by Zhang et al. in 2020 and has been shown to correlate well with human judgments for many text-generation tasks. BERTScore leverages the pre-trained embedding from BERT and matches words in generated and reference sentences by cosine similarity. Moreover, BERTScore considers both precision and recall and combines them to compute an F_1 measure, which is appropriate for evaluating generated feedback in our task. Thus, we employ BERTScore to measure semantic similarity between expert feedback and generated feedback.

Human-centered evaluation: Human-centered evaluation asks human evaluators to judge the quality of generated text along some specific dimensions (e.g., readability). Caligiuri and Thomas [6] found that a positive tone and suggestions for improvement are key features of good reviews. Jia et al. [20] mentioned that providing suggestions, mentioning problems, and using a positive tone, are main characteristics of effective feedback. Celikeyilmaz et al. [9] pointed out that fluency and factuality are essential for evaluating system-generated text. In this paper, we manually evaluate generated feedback in five dimensions: Readability, Suggestions, Problems, Positive Tone, and Factuality. The details of these evaluation dimensions are described in Section 4.5.3.

2.3 Ethical Concerns in Feedback Generation

To date, ethical issues in feedback generation have received scant attention in the research literature. Celikyilmaz et al. [9] pointed out that a potential ethical issue that may appear in text-generation tasks is the problem of generating improper or offensive language. Li et al. [24] also mentioned that using pre-trained language models for text-generation tasks may raise ethical issues, such as generating private content, because corpora used for pre-training are pulled from the web without carefully filtering. However, there is still a lack of systematic methods for evaluating how effectively a system can avoid generating inappropriate text. For these reasons and others, this work considers the ethical implications of an automated feedback system on student project reports by manually inspecting all generated feedback.

Table 1: Sample Human Reference Reviews

Very good writeup, as far as it goes. Good discussion of test cases and reasons for refactoring. It would have helped to see some code snippets. Good section on Future Refactoring Opportunities.
A very good description of changes, and appropriate code snippets are shown. Manual testing is shown with annotated screenshots, which are very useful. In a Rspec test, sending emails to an actual person's address is not a good practice.
The wiki page covers all necessary items, but the "test plan" part can be more elaborated. And the last screenshot is useless. It will be better to show the DB records, instead of table structure.

Table 2: Statistics on the Dataset. We measure the percentage of reports that contain more than 1024 tokens since it is the maximum input length limit of the BART model, and we utilize BART to craft the generation function in our system.

# of data samples = 484		Average	Percentile					Maximum	Over 1024 tokens
			5th	25th	50th	75th	95th		
Reports	# of words	1193	405	734	1060	1499	2374	6512	
	# of subword tokens	1643	583	1040	1489	2025	3272	8422	76 %
Summarized Reports	# of words	704	395	649	742	803	865	908	
	# of subword tokens	951	580	1004	1021	1023	1024	1024	0 %
Reference Reviews	# of words	55	13	33	48	72	114	258	
	# of subword tokens	71	19	43	61	90	147	335	-

3. DATA

In this section, we introduce a new dataset collected for this study. Firstly, we describe the data source in Section 3.1. Then, we explain in Section 3.2 how participants' privacy rights were respected during the data collection process. Finally, we present statistics on the dataset in Section 3.3.

3.1 Data Source

The data used in this study are collected from a graduate-level object-oriented development course at a public university in the United States. For a course assignment, two to four students form a group and work together on some course project, bid on topics from a list of potential projects provided by course instructors. The dataset includes group projects where students refactor code from an open-source project Expertiza, add new features to it, or write automated tests for a software module that needs them.

As a part of project deliverables, each team is required to submit a group report to document the work that has been completed, methodologies that they have utilized, and other project-related material (e.g., how they test their code). Such group reports are also called wiki pages since they are added to the wiki document maintained by the open-source project Expertiza. The instructor reviews each of these wiki pages (i.e., each group report) and provides textual feedback on each of them. To better understand our data, URLs to three anonymous and de-identified reports are provided in footnotes[1][2][3]. Three randomly sampled[4] instructor reviews from our dataset are displayed in Table 1.

[1]Sample report 1: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%201.pdf`

[2]Sample report 2: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%202.pdf`

[3]Sample report 3: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%203.pdf`

[4]According to our IRB protocol, in order to protect participants' privacy rights, we do not display instructor reviews for the sample group reports.

3.2 Privacy protection

In this work, we take our responsibility to protect the privacy of students' data very seriously. The use of the dataset has been approved by the IRB at our institution. Sensitive student data was de-identified and handled in a way that is FERPA compliant. More specifically, our data protection and de-identification procedure consist of four main steps: 1) we took our data from an anonymized database, which uses random identifiers for students and groups; 2) we utilized regular-expression techniques to automatically remove all names from reports; 3) we manually inspected and removed remaining sensitive data, such as links to documents that might identify individual authors; 4) we stored data securely on a cloud drive managed by the university.

3.3 Statistics on the Dataset

We collected de-identified students' project reports and associated textual feedback provided by instructors from twelve semesters between Spring 2015 and Spring 2021. This gave us a set of 484 group projects. Table 2 summarizes statistics from the dataset. Note that since the pre-trained language model BART (detailed in Section 4) has a maximum input length limit of 1024 tokens[5], we employ an unsupervised method to summarize original reports to lengths acceptable for input into the pre-trained BART. Statistics on these summarized reports are provided in the third row of the table.

We measured the average number of words and tokens for reports, summarized reports, and expert reviews in our dataset. The average number of words for each original report is 1193, which corresponds to 1643 subword tokens. We found that 75.8% of the reports comprise more than 1024 tokens. The average number of words in each summarized report is 704 (equivalent to 951 subword tokens). For expert reviews, the average number of words and the average number of tokens per instructor review are 55 and 71, respectively.

[5]A token is an instance of a sequence of characters that are grouped together as a useful semantic unit for processing. It is similar to, but not identical with, morpheme.

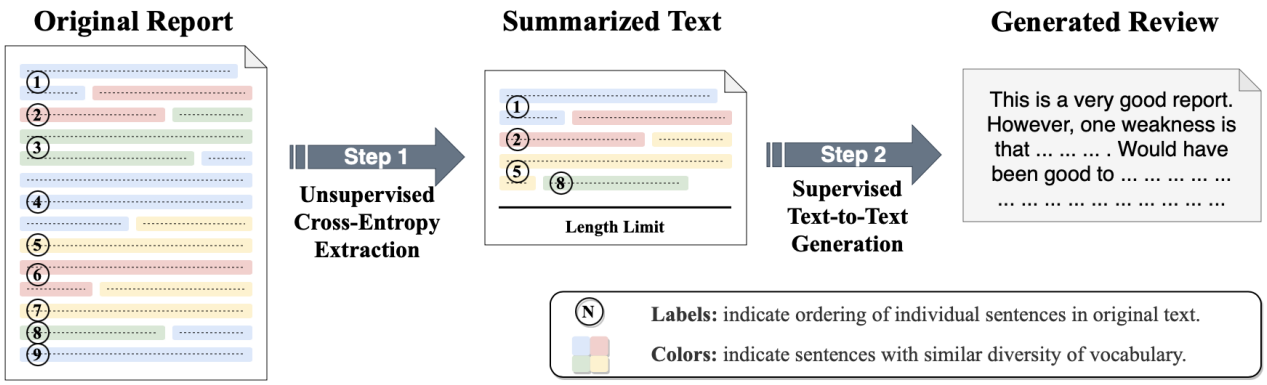

Figure 1: Operation of our *Insta-Reviewer* System. The system uses a select-then-generate paradigm [44, 18, 52]. The first step is to extract salient sentences (within the length limit) from students' project reports. The second step is to utilize a supervised NLP model to generate feedback for students' reports. Details of the system are described throughout Section 4.

4. METHODOLOGY

In this section, we detail our data-driven approach for automatically generating feedback on students' project reports. We first formally define our task in Section 4.1. Then, we present the overall design of our feedback-generation system in Section 4.2. After that, Section 4.3-4.4 elaborates all components of the system. A set of metrics for evaluating generated textual feedback is given in Section 4.5. Finally, in Section 4.6, some ablation experiments are proposed to understand the contribution of each component.

4.1 Problem Formulation

We formulate the task of automatic feedback generation for students' project reports as a *text-to-text generation* problem, where the source text is a long project report and the target text is a review. Our dataset can be represented as $D = (X^i, Y^i)_{i=1}^{N=484}$, where $X^i = \langle x_1^i, ..., x_j^i, ..., x_n^i \rangle$ denotes a sequence of input tokens representing an instance of report, and $Y^i = \langle y_1^i, ..., y_k^i, ..., y_m^i \rangle$ denotes a sequence of output tokens representing the corresponding textual feedback. Each token x_j^i or y_k^i is drawn from a token vocabulary $\mathcal{V}$.

Then, the problem can be formally described as:

$$Y = \mathcal{F}_{\mathcal{M}}(X, \mathbb{C}) \tag{1}$$

where the model, or generation function, $\mathcal{F}_{\mathcal{M}}$ takes a sequence of tokens X (i.e., a project report) as the input, and produces a sequence of tokens Y (i.e., generated feedback for the project report) as the output, while satisfying a set of constraints $\mathbb{C}$, which is a collection of desired properties (e.g., fluency, coherence, and length) for the output text.

The objective of the task is to effectively model the generation function $\mathcal{F}_{\mathcal{M}}$ in a data-driven manner using the dataset, so that it can generate plausible and readable feedback for unseen reports. In this work, the generation function $\mathcal{F}_{\mathcal{M}}$ is crafted based on a pre-trained language model (PLM) called BART (detailed in Section 4.4), which has been demonstrated to be the state-of-the-art method to model the generation function for various text-to-text generation tasks [23].

4.2 System Design

Despite the fact that the pre-trained language model BART is an effective method to model the generation function $\mathcal{F}_{\mathcal{M}}$, it has an input length limit of 1024 tokens[6] [23]. Nevertheless, group reports are usually longer than that. In our dataset, 75.8% of the reports contain more than 1024 tokens, and the longest report is approximately eight times longer than that limit. One simple fix is to truncate the report by discarding all tokens beyond the length limit, but this can cause loss of critical information from the inputs (this hypothesis is verified in Section 4.6.1). Thus, we adopt a select-then-generate method [18, 44, 52] to generate feedback on student project reports, as illustrated in Figure 1.

Overall, the select-then-generate paradigm decomposes the problem into two sequential subproblems to resolve the issue of overlength input documents: 1) an unsupervised sentence-level extractive summarization task, and 2) a supervised PLM-based text-to-text generation task. More formally, the problem description becomes,

$$Y = \mathcal{F}_{\mathcal{M}}(\mathcal{S}_{\mathcal{E}}(X), \mathbb{C}) \tag{2}$$

where the input to the generation function $\mathcal{F}_{\mathcal{M}}$ becomes $\mathcal{S}_{\mathcal{E}}(X)$, which represents a summarized report. The new function $\mathcal{S}_{\mathcal{E}}$ represents an extractive summarizer, which can effectively extract salient sentences from an original report X and produce a summarized version of the report as the input to the feedback-generation function $\mathcal{F}_{\mathcal{M}}$.

Thus, our automated feedback-generation system comprises two stages. In the first stage, we use an unsupervised method, called cross-entropy extraction, to summarize original reports to lengths acceptable for input into the PLM BART (i.e., the implementation of the generation function $\mathcal{F}_{\mathcal{M}}$). In the second stage, we train the PLM BART on the summarized reports $\mathcal{S}_{\mathcal{E}}(X)$ and reference reviews Y. In the following sections (Section 4.3-4.4), we detail each stage of our feedback-generation system *Insta-Reviewer*.

[6]The length is limited to 1024 since the BART authors [23] chose this number and pre-trained the model with this limit.

4.3 Step 1: Cross-Entropy Extraction

The goal of the first step is to summarize original overlength reports to lengths acceptable for input into the pretrained language model (PLM) BART that will be used in the second step, while retaining as diverse a subset of content as possible. We adopt an unsupervised extractive-summarization method, called cross-entropy extraction [15].

The cross-entropy method is an unsupervised technique that treats sentence-level extractive summarization as a combinatorial optimization problem [42]. More formally, we can let S_D denote the set of all sentences in the report we are trying to summarize. From this set we want to extract a subset of sentences $S \subset S_D$ that maximizes some quality target function $Q(S)$. This quality target function can be comprised of whatever features and measures deemed applicable to the task at hand. In our case, we used only a single feature in our quality function, namely the diversity of vocabulary in the summary. The reasoning behind this is that sentences with a more diverse vocabulary will also cover a more diverse set of information from the text. To measure this diversity explicitly, we calculate the unigram LM entropy of the summary S, as shown below,

$$Q(S) \;=\; -\sum_{w \in S} p_S(w) \log p_S(w) \tag{3}$$

$$p_S(w) = \frac{\textsc{Count}(w)}{\textsc{Len}(\mathrm{S})} \tag{4}$$

Note that in the above equations w represents a single word in the summary S. Additionally, the function $\textsc{Count}(w)$ measures the frequency of the word w in summary S and $\textsc{Len}(\mathrm{S})$ is the total number of words in the summary. Additionally, to encourage the method to prefer summaries as close to the length constraint as possible, we added an additional term to this quality function that is proportional to the number of tokens in S, denoted $\textsc{Tokens}(\mathrm{S})$, the intuition behind this being that the BART model will perform better in most cases when it has more text to work with. The final quality function with this added length term is shown below, where β, our proportionality constant, is a hyper-parameter of the model.

$$Q(S) \;=\; \beta \cdot \textsc{Tokens}(\mathrm{S}) - \sum_{w \in S} p_S(w) \log p_S(w) \tag{5}$$

In order to enforce the length constraint on our summaries, we simply assign $Q(S) = -\infty$ whenever S has a length of greater than 1024 BART tokens. The actual output of the cross-entropy method is a vector $p = \langle p_1, p_2, \ldots, p_n \rangle$ indicating the probability of selection in the summary for each of the n sentences in the original document. Initially, these probabilities start out the same for each sentence, but they quickly convert to either 0, for sentences that result in low Q values, or to 1, for sentences that result in high Q values. Below are the steps of the algorithm we used for carrying out this cross-entropy extraction procedure.

1. **Preprocessing Text**: For each student project report, we split it into its n sentences, enumerating each according to their order in the report. We then tokenize each sentence into word tokens, being sure to remove all stop words, punctuation, etc., when doing so.
2. **Initialize** p: Initially we want each sentence to be equally likely to be chosen, so set $p_0 = \langle 1/2, 1/2, \ldots, 1/2 \rangle$. If $n > 60$ then we reduce this probability to ensure a sufficient sample of summaries that meet our length constraint. Then we set $t = 0$.
3. **Sample Summaries**: We sample N Bernoulli vectors, X_1, X_2, …, X_N according to the probability vector p_{t-1}. The sentences selected from each of these vectors define our N sample summaries S_1, S_2, …, S_N. Set t = t+1.
4. **Quality Scores**: For each of the summaries S_i, we calculate its quality performance score according to the above equations. We determine the cutoff value of the elite sample, γ_t, which is the Q value of the $(1-\rho)$ sample quantile.
$$\gamma_t \;=\; Q(S)_{\lceil (1-\rho)N \rceil} \tag{6}$$
5. **Update** p: We use the sample values to update our probabilities, storing them as $\hat{p}_t$, according to the update rule below:
$$\hat{p}_{t,j} \;=\; \frac{\sum_{j=1}^{N} \delta_{[Q(S_j) \geq \gamma_t]} \delta_{[X_{i,j}=1]}}{\sum_{j=1}^{N} \delta_{[Q(S_j) \geq \gamma_t]}} \tag{7}$$
where $\delta_{[c]}$ is the Kronecker-delta function which evaluates to 1 if the condition c is satisfied, otherwise 0.
6. **Smooth** p: To balance exploration and exploitation of the summary samples, we smooth p_t like so:
$$p_t \;=\; \alpha \hat{p}_t + (1-\alpha) p_{t-1} \tag{8}$$
7. **Termination**: If the value of γ_t has not changed in 3 iterations then the process terminates, returning the current p_t. Otherwise, it returns to step 3 and repeats.

In our implementation of the above algorithm, we found the following parameter settings to be optimal: $N = 1000$, $\rho = .05$, $\alpha = .7$. To get our final truncated summary text, we simply sample one more Bernoulli vector, X, using the final sentence extraction probabilities p. After doing so, we check that the resulting summary defined by the sentences in X meets our length constraint. If it does not, then we would sample a new Bernoulli vector X until we found a summary that did (though this was never necessary).

4.4 Step 2: PLM-based Generation Model

We now describe the second step of the approach. The objective of the second step is to effectively craft the feedback-generation function $\mathcal{F}_{\mathcal{M}}$ in Equation 2. We first introduce the BART model used for crafting $\mathcal{F}_{\mathcal{M}}$, then we describe the beam-search method for improving the performance.

4.4.1 Modeling the Generation Function with BART

In this work, we employ a state-of-the-art PLM BART [23], which stands for bidirectional and auto-regressive transformers, to model the generation function $\mathcal{F}_{\mathcal{M}}$. The BART model is suitable for text-to-text generation tasks since it utilizes an encoder-decoder architecture (as illustrated in Figure 2), which can effectively model complex mappings (i.e., the underlying patterns) from one sequence of text (e.g., summarized reports) to another (e.g., feedback).

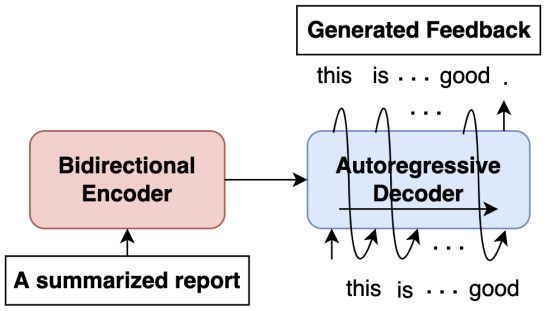

Figure 2: Illustration of an encoder-decoder architecture. The encoder can convert an input sequence of text (e.g., a summarized report) into a rich numerical representation, and then the decoder generates the output sequence (e.g., feedback) by iteratively predicting the most probable next word.

The BART framework consists of two steps: pre-training and fine-tuning. Instead of training the model from scratch on our dataset, the BART model is first pre-trained on a large generic corpus over different pre-training tasks, and then all parameters of the model are fine-tuned on our data. The model can acquire a sophisticated "understanding" of human grammar through the pre-training process, thus significantly reducing the need for annotated data when training the feedback-generation model while improving convergence rate and generalization [13]. In this work, we use the pre-trained checkpoint "facebook/bart-large-cnn[7]" to initialize all parameters of the model and then fine-tune the model on a training set drawn from our dataset D.

4.4.2 Diverse beam search for decoding

After fine-tuning the BART model on our data, we use the diverse beam search (DBS) [46] algorithm to decode the output sequences in inference time to generate better feedback. In the original BART model setting, feedback is generated by iteratively selecting the word with the highest probability at each position in the sequence, which is referred to as greedy decoding. Greedily choosing the word with the highest probability at each step might be optimal at the current spot in the sequence, but as we move through the rest of the full sentence, it might turn out to be a non-optimal choice (output can be ungrammatical, unnatural, and nonsensical) since the greedy decoding algorithm lacks backtracking.

On the other hand, the DBS algorithm keeps track of the top-n most probable next words, where n is the number of beams. The next set of beams is chosen by considering all possible next-word extensions of the existing set and selecting the n most likely extensions. Additionally, in order to improve the diversity in the outputs, all beams are divided into groups, and diversity between the groups is enforced by the DBS algorithm. In our experiments, we set beam size to 4 and the number of groups to 2 since this combination yields the best results. DBS is applied to all models in this work, including the BigBirdPegasus model (Section 4.6.2).

4.5 Evaluation Metrics

[7]https://huggingface.co/facebook/bart-large-cnn

We evaluate generated feedback with a comprehensive set of metrics, including a content-overlap metric ROUGE, a model-based metric BERTScore, and a new human-evaluation metric. As previously mentioned, the ultimate goal of InstaReviewer is to generate feedback that is helpful to students instead of generating the same feedback as provided by instructors. To this end, human-centered evaluation is considered the gold standard. ROUGE and BERTScore are employed to validate our human-evaluation results since human evaluations may be inconsistent and subjective, which can lead to erroneous conclusions [9]. The intuition is that while instructor feedback may only focus on certain aspects and be imperfect, it is valuable to know how similar the generated feedback is to the feedback provided by instructors. The implementations of the metrics are described below.

4.5.1 Content-overlap Metric: ROUGE

We use the standard ROUGE metric to measure content overlap between generated feedback and expert feedback. Specifically, we report the F_1 scores for ROUGE-1, ROUGE-2, and ROUGE-Lsum, respectively measuring the word-overlap, bigram-overlap, and longest common sequence between the texts. We obtain our ROUGE scores using the Google rouge package[8] [25, 16]. Porter stemming is enabled to remove plurals and word suffixes (e.g., "ing", "ion", "ment").

4.5.2 Model-based Metric: BERTScore

The BERTScore metric is leveraged to assess the semantic equivalence between generated feedback and expert feedback. We report the F1 measure of BERTScore that combines both precision and recall, which is proper for evaluating generated feedback in our task. We calculate the BERTScore utilizing the official BERTScore script[9] [55].

4.5.3 Human Evaluation

After reviewing relevant papers (e.g., [6, 20, 50, 57]) and discussions among the authors of this paper, we selected the following five dimensions to evaluate the feedback manually: Readability, Suggestions, Problems, Positive Tone, and Factuality. Our scores for these five manual-evaluation dimensions are calculated as follows:

(i) ***Readability*** (READ): In this work, readability is defined as the quality of feedback in grammar, word choice, and coherence. We judge it using a five-point rating scale: 0. Incomprehensible 1. Not fluent and incoherent 2. Somewhat fluent but incoherent 3. Fluent but somewhat incoherent 4. Fluent and coherent.

(ii) ***Suggestions*** (SUGG): Providing suggestions is a key feature of quality feedback that is valuable to students. We give a score of 1 if the feedback contains at least one valid suggestion statement that can guide the reviewee in how to correct a problem or make improvements. Otherwise, we give a score of 0.

(iii) ***Problems*** (PROB): Pointing out something that is going wrong in students' work is also important for helping learners. We give a score of 1 if the feedback describes at least one issue that needs to be addressed in the student report. Otherwise, we give a score of 0.

[8]https://github.com/google-research/google-research/tree/master/rouge

[9]https://github.com/Tiiiger/bert_score

(iv) ***Positive Tone*** (TONE): Feedback phrased in a positive tone can better stimulate students' reflective competence. We give feedback a score of 1 if it has an overall positive semantic orientation, 0.5 if it is neutral, and 0 if it is negative.

(v) ***Factuality*** (FACT): The statements in generated feedback should be factually correct. Otherwise, they may inadvertently mislead the reviewee and negatively impact learning. Factuality is calculated as:

$$\text{FACT} = \frac{\text{Count(factually correct statements)}}{\text{Count(total statements)}}$$

where the numerator is the total number of factually correct statements, and the denominator is the total number of statements in the feedback. If the denominator is 0, we directly give a score of 0.

This set of evaluation criteria is certainly not perfect, but it balances the accuracy and cost of the evaluation. For example, we could score the "Problems" dimension in a more sophisticated and accurate way, but it would be very time-consuming and expensive. We leave more accurate and efficient human-evaluation criteria for future work.

4.6 Ablation experiments

In order to understand the contribution of each step in our method, we designed two ablation experiments:

4.6.1 Ablation Exp. 1 - Naïve BART:

In the first ablation experiment, we intend to understand the contribution of the cross-entropy (CE) summarization. As we mentioned in Section 4.2, a straightforward way to solve the BART's length-limit problem is to truncate all tokens beyond the length limit, which we hypothesize may lead to a loss of critical information from the inputs. Therefore, we measure the performance of a BART model that simply truncates all tokens exceeding the length limit of 1024 as input, and we call this setup - "Naïve BART." If adequate information is contained in the truncated reports to generate feedback on students' reports, then "CE + BART" should perform similarly to "Naïve BART." Otherwise, it would demonstrate that using CE to summarize the original students' project reports is better than simply truncating all tokens that exceed the length limit. In other words, it will suggest that the CE method can effectively summarize students' reports while retaining critical information that helps the model generate feedback, and the CE step is necessary.

4.6.2 Ablation Exp. 2 - BigBirdPegasus:

In the second ablation experiment, we investigate whether the "CE + BART" method can be replaced using the recently proposed sparse-attention-based PLM BigbirdPegsus [54], which extends the input length limit to 4096 tokens. Briefly, BigBirdPegasus increases the input length limit at the cost of using a sparse attention mechanism, which may reduce the ability of the model to fit complex mappings between texts. Similar to BART, BigBirdPegasus also consists of the pre-training and fine-tuning steps. In this experiment, we use the pre-trained checkpoint "google/bigbird-pegasus-large-arxiv[10]" to initialize all parameters of BigbirdPegsus and then fine-tune the model on our data.

[10]https://huggingface.co/google/bigbird-pegasus-large-arxiv

5. EXPERIMENTS AND RESULTS

5.1 Experimental Setup

5.1.1 Training Details

For all experiments, we train our models with a batch size of 1/2, a learning rate of 2e-5/3e-5/5e-5, epochs of 2/3, and the AdamW optimizer [26] with a weight decay of 0.01 and a linear rate scheduler of 10% warm-up steps. For our dataset D, we use an 80-10-10 split for training, validation, and test data. After finding the optimal hyper-parameters, we merge the training and validation sets into the new training data.

5.1.2 Hardware Setup

The BART models are trained on an NVIDIA RTX6000 GPU (24GB). The BigBirdPegasus model (mentioned in Section 4.6.2) is trained on an NVIDIA A6000 GPU (48GB). We also employ the automatic mixed-precision training (use of both 16-bit and 32-bit floating-point types) to speed up the training processes.

5.2 Experimental Results

To investigate the potential promise of our system, we design experiments to answer the following research questions:

RQ1: How effective is the proposed approach for generating feedback on students' project reports?

The goal of our first experiment is to find out if our automated feedback system Insta-Reviewer is effective in generating some readable and helpful feedback for students' project reports. The evaluation results of Insta-Reviewer are shown in Table 3. Some generated feedback for actual student reports is shown in Table 5, and their evaluation scores are shown in Table 4. According to Table 3, the ROUGE1, ROUGE2, ROUGELsum, and BERTScore for our "CE + BART" method are 28.54, 6.39, 18.21, and 59.18, respectively. These results imply that the generated and expert feedback are basically consistent in semantics, and there is some overlap in words (more precisely, n-grams).

We also evaluated the feedback provided by instructors, and the first row shows the human-evaluation scores for it. Compared to the expert feedback, we surprisingly find that in terms of "Problems" and "Positive Tone," our method can outperform human experts by 6% and 2%. However, it is worth noting that the generated feedback tends to mention more generic problems (e.g., missing a test plan) rather than project-specific issues (e.g., "xxx files should be described"). Additionally, the expert feedback can outperform the generated feedback, with gaps of 6%, 16%, 15.2% for "Readability," "Suggestions," and "Factuality," respectively. In summary, although our system is not as good as experts at providing suggestions and may produce some non-factual statements, it is good at generating fluent and positive feedback that mentions problems that need to be addressed.

RQ2: What are the problems of system-generated reviews? What are they not good at?

In the second experiment, we aim to explore the potential issues in system-generated feedback in more detail. Although the vast majority of the feedback is fluent, positive, and factually correct, we find two potential problems after manually inspecting all system-generated feedback in the test set.

Table 3: ROUGE F_1 (with porter stemming), BERTScore, and Human Evaluation scores (%) on the test set (n=50). All our ROUGE scores have a 95% confidence interval (CI) of at most ±2.98 as reported by the official ROUGE script (bootstrap resampling). Our BERTScore has a 95% CI of at most ±1.54. The BERTScore script does not support BigBirdPegasus currently.

	Method	ROUGE			BS	Human Evaluation					
		R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT	Average
	1. Expert Feedback	-	-	-	-	**100.0**	**82.0**	90.0	93.0	**100.0**	**93.00**
Insta-Reviewer	2. CE + BART	**28.54**	**6.39**	**18.21**	**59.18**	94.0	66.0	**96.0**	**95.0**	84.8	87.16
Ablation Exp 1	3. Naïve BART	25.88	4.98	17.23	58.03	82.0	58.0	94.0	93.0	76.8	80.76
Ablation Exp 2	4. BigBirdPegasus	15.92	4.04	13.45	-	56.0	2.0	12.0	58.0	23.4	30.28

Table 4: ROUGE F_1 (with porter stemming), BERTScore, and Human Evaluation scores (%) for the sample generated feedback shown in the Table 5. †"I would suggest that the reader read the ..." is not a valid suggestion for the reviewee.

	ROUGE			BERTScore	Human Evaluation					
	R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT	Average
Generated Feedback 1	27.52	1.87	20.18	59.17	100.0	100.0	100.0	100.0	100.0	100.0
Generated Feedback 2	45.09	18.00	21.57	66.79	100.0	0.00†	100.0	100.0	83.33	76.67
Generated Feedback 3	22.22	0.00	13.89	52.92	80.00	0.00	100.0	100.0	80.00	72.00

Table 5: Sample expert and generated feedback. Scores for the generated feedback are shown in Table 4. The generated feedback may contain non-factual or ambiguous statements.

Expert Feedback 1: "The page is pretty well organized in what changes they made. Their pictures were not taken as a screenshot, instead were just taken with their phone camera. They didn't provide a manual test plan, which is problematic as we have no good way of verifying their changes without significant work involved."

Generated Feedback 1: "The writeup is quite readable, and the changes are described in good detail. However, the code changes are not described in prose, and there is no explanation of how they are implemented. The test plan is not described, and it would be much more useful to have a description of the tests that are to be added."

Expert Feedback 2: "Your examples say that code was added, but don't describe the code or what it is used for. The reader needs to figure that out by reading the code. For someone following on, it would be much more useful to have a description of how the code works. "

Generated Feedback 2: "The writeup is very readable, but it does not explain the changes made to the code. It is not clear how the code was changed. The test plan is not described. There is no description of the tests. I would suggest that the reader read the code to figure out what they test."

Expert Feedback 3: "Well written, but does not mention design patterns or principles. 'Gemfiles' should be downgraded. Schema changes should be shown as migrations and not direct changes in schema file."

Generated Feedback 3: "The wiki page is very short, but the changes are described well. It is not clear why the changes were made. The code changes are not described in prose. They (← code snippets) are just shown. I would have liked to see more explanation of the changes"

Firstly, the automated feedback system may occasionally (≈15.2% of all statements) generate some non-factual or ambiguous statements in the feedback. For instance, the "generated feedback 2" shown in Table 5 contains a non-factual statement (italicized): "... there is no description of the tests. *I would suggest that the reader read the code to figure out what they test.*" However, this statement should probably be "... a description of the tests needs to be added. Otherwise, readers need to read the code to figure out what you test."

Secondly, some pieces of text frequently appear in different generated feedback. For instance, we find that 66% of the system-generated feedback contains the sentence "the writeup is very/quite readable." We speculate that this happens because 14% of the expert feedback that used for training contains the exact same sentence, which introduces some sort of "imbalance" problem. This repetition problem is not necessarily a drawback of the system, but it suggests that high-frequency text pieces can influence the generation.

RQ3: How does the system perform compared with other potential methods?

We now turn our attention to the experimental results of the two ablation experiments detailed in Section 4.6. The evaluation results are presented in Table 3. The second row, the third row, and the fourth row show the performance scores for our Insta-Reviewer system, naïve BART (the method utilized for the ablation experiment 1), and BigBirdPeagsus (the model used for the ablation experiment 2), respectively.

The goal of the first ablation experiment is to verify our hypothesis that using the cross-entropy extraction (CE) to summarize the original students' project reports is better than simply truncating all tokens beyond the length limit of BART. By looking at the ROUGE scores and BERTScore, we can find that the "CE + BART" method consistently outperforms the "naïve BART" method, with gaps of 2.66, 1.41, 0.98, and 1.15 for R1, R2, RLsum, and BERTScore

Table 6: ROUGE F_1 (with porter stemming) and BERTScore on the test set (n=50) in small-data settings ($\leq$ 100 samples). For each score, we report a 95% confidence interval (CI).

data points	ROUGE			BERTScore
	R1	R2	RLsum	
434	28.54±2.98	6.39±1.56	18.21±1.83	59.18±1.54
50	24.29±2.47	4.99±0.94	17.42±1.70	56.12±1.42
100	25.75±2.65	5.56±1.15	17.86±1.61	57.36±1.43

respectively. Additionally, based on the human evaluation scores, "CE + BART" significantly ($\geq$ 8%) outperforms the "Naïve BART" method on "Readability," "Suggestions," and "Factuality." The "CE + BART" method can also achieve higher ($\geq$ 2%) scores on dimensions "problems" and "positive tone." The results demonstrate that the CE method can effectively summarize students' reports while retaining critical information that helps the model generate feedback.

The second ablation experiment aims to know whether the sparse-attention-based PLM BigBirdPeagsus can effectively fit the complex mappings from student reports to textual feedback and replace our "CE + BART" approach. As shown in Table 3, the results clearly indicate that "CE + BART" substantially outperform the BigBirdPeagsus model on all metrics. The results suggest that although the BigBirdPeagsus model extends the input length limit to 4096 by using the sparse-attention mechanism, it may not be suitable for complicated tasks such as generating feedback.

RQ4: How does the automated feedback system perform in different small-data settings?

In order to help other researchers who intend to apply our approach on their datasets, we evaluated the effectiveness of the approach in different small-data settings. Table 6 shows the ROUGE scores and BERTScore when training with only 50 and 100 data samples, respectively. The scores (R1=25.75, R2=5.56, RLsum=17.86, BS=57.36) for our model when training with 100 samples are similar to the scores (R1=25.88, R2=4.98, RLsum=17.23, BS=58.03) for the "Naïve BART" when training with 434 samples. The results demonstrate that the performance of our approach is acceptable in small-data settings. Thus, our method can potentially be applied to other tasks that have limited data.

RQ5: Does the Insta-Reviewer automatic feedback system raise any ethical concerns?

The objective of the last experiment is to investigate whether the automated feedback system raises any ethical concerns. In this work, we consider two main potential ethical issues related to the system. The first major concern is whether the system will generate improper or offensive language. The second potential issue is whether the generated feedback will contain some private content. Although we have filtered out all inappropriate information from our dataset, these ethical concerns may still arise since the BART model is pre-trained on a large-scale corpus crawled from the Internet without fine-grained filtering. Therefore, we manually vetted all generated feedback, and failed to find any of the aforementioned ethical transgressions.

6. CONCLUSION

Timely feedback is critical to learning because it is more likely to motivate students to stay on task and achieve their learning goals. This suggests that future AI-powered educational applications will include automated feedback systems to generate real-time feedback. In this paper, we have presented a data-driven system, named *Insta-Reviewer*, for generating instant textual feedback on students' project reports. The system leverages a select-then-generate paradigm consisting of two main steps: 1) cross-entropy extraction and 2) BART-based supervised text-to-text generation. The results demonstrate that the generated feedback could achieve near-human performance and even outperform human experts in the "Problems" and "Positive Tone" dimensions. However, the system may occasionally generate some non-factual or ambiguous statements in the feedback. The generated feedback seems to be free of any ethical complications. Our work demonstrates the feasibility of automatic feedback generation for students' project reports while laying the groundwork for future research on this topic.

Limitations: There are three main limitations to this study. Firstly, we simply used textual information extracted from the student reports and ignored all images. As a result, our model could not produce feedback like "Their pictures were not taken as a screenshot, instead were just taken with their phone camera." If we could design a multi-modality model that incorporates all text, images, and artifacts such as code into the input, we would be able to provide more comprehensive feedback to students. Secondly, we used a set of metrics, including ROUGE, BERTScore, and human-evaluation scores to evaluate the generated feedback. However, ROUGE and BERTScore cannot accurately reflect the quality of the generated feedback. Human evaluation can more accurately assess the quality, but it is inconsistent, subjective, and time-consuming. Thus, we believe it is worthwhile to explore more effective automatic metrics to evaluate the generation. Thirdly, we manually inspected all generated feedback, and found that it did not raise any ethical concerns. Nevertheless, this result does not guarantee that the model will never produce improper or offensive language. Systematic methods should be investigated to evaluate how the system can avoid generating inappropriate language. However, this problem is particularly challenging because the output of neural networks is not always predictable.

Future Work: An important direction for future work is to investigate how to avoid generating non-factual statements in feedback. This problem of factual correctness has two potential solutions. The first promising way is to automatically evaluate the correctness of each sentence in generated feedback and remove all non-factual statements before delivering the feedback to students. Dusek & Kasner [12] have proposed an entailment-based model to evaluate the correctness of the generated text. However, this approach does not capture which part of the generated text is non-factual. Future work can explore further along this direction. The other possible method to address the problem of non-factual statements is to design new architectures that can more effectively capture the complex mappings from student reports to feedback. The latter approach is significantly more challenging but may get to the root of the problem.

7. REFERENCES

[1] W. Alharbi. E-feedback as a scaffolding teaching strategy in the online language classroom. *Journal of Educational Technology Systems*, 46(2):239–251, 2017.

[2] M. Ariely, T. Nazaretsky, and G. Alexandron. First steps towards nlp-based formative feedback to improve scientific writing in hebrew. *International Educational Data Mining Society*, 2020.

[3] 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*, 1:391–402, 2013.

[4] M. Borowczak et al. Communication in stem education: A non-intrusive method for assessment & k20 educator feedback. *Problems of Education in the 21st Century*, 65(1):18–27, 2015.

[5] R.-M. Botarleanu, M. Dascalu, M.-D. Sirbu, S. A. Crossley, and S. Trausan-Matu. Readme–generating personalized feedback for essay writing using the readerbench framework. In *Conference on Smart Learning Ecosystems and Regional Development*, pages 133–145. Springer, 2018.

[6] P. Caligiuri and D. C. Thomas. From the editors: How to write a high-quality review, 2013.

[7] C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluating the role of bleu in machine translation research. In *11th conference of the european chapter of the association for computational linguistics*, pages 249–256, 2006.

[8] D. Carless, D. Salter, M. Yang, and J. Lam. Developing sustainable feedback practices. *Studies in higher education*, 36(4):395–407, 2011.

[9] A. Celikyilmaz, E. Clark, and J. Gao. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*, 2020.

[10] G. Deeva, D. Bogdanova, E. Serral, M. Snoeck, and J. De Weerdt. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education*, 162:104094, 2021.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] O. Dušek and Z. Kasner. Evaluating semantic accuracy of data-to-text generation with natural language inference. *arXiv preprint arXiv:2011.10819*, 2020.

[13] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[14] D. J. Evans, P. Zeun, and R. A. Stanier. Motivating student learning using a formative assessment journey. *Journal of anatomy*, 224(3):296–303, 2014.

[15] G. Feigenblat, H. Roitman, O. Boni, and D. Konopnicki. Unsupervised query-focused multi-document summarization using the cross entropy method. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 961–964, New York, NY, USA, 2017. Association for Computing Machinery.

[16] K. Ganesan. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*, 2018.

[17] E. F. Gehringer. A course as ecosystem: melding teaching, research, and practice. In *2020 ASEE Virtual Annual Conference Content Access*, 2020.

[18] S. Gehrmann, Y. Deng, and A. M. Rush. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*, 2018.

[19] J. Hattie and H. Timperley. The power of feedback. *Review of educational research*, 77(1):81–112, 2007.

[20] Q. Jia, J. Cui, Y. Xiao, C. Liu, P. Rashid, and E. F. Gehringer. All-in-one: Multi-task learning bert models for evaluating peer assessments. *arXiv preprint arXiv:2110.03895*, 2021.

[21] J. A. Kulik and C.-L. C. Kulik. Timing of feedback and verbal learning. *Review of educational research*, 58(1):79–97, 1988.

[22] S. Kusairi. A web-based formative feedback system development by utilizing isomorphic multiple choice items to support physics teaching and learning. *Journal of Technology and Science Education*, 10(1):117–126, 2020.

[23] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[24] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen. A survey of pretrained language models based text generation. *arXiv preprint arXiv:2201.05273*, 2022.

[25] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[26] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[27] C. Lu and M. Cutumisu. Integrating deep learning into an automated feedback generation system for automated essay scoring. *International Educational Data Mining Society*, 2021.

[28] A. Malik, M. Wu, V. Vasavada, J. Song, M. Coots, J. Mitchell, N. Goodman, and C. Piech. Generative grading: Near human-level accuracy for automated feedback on richly structured problems. *arXiv preprint arXiv:1905.09916*, 2019.

[29] A. Mallavarapu. Exploration maps, beyond top scores: Designing formative feedback for open-ended problems. In *Proceedings of The 13th International Conference on Educational Data Mining*, 2020.

[30] E. Mannix and M. A. Neale. What differences make a difference? the promise and reality of diverse teams in organizations. *Psychological science in the public interest*, 6(2):31–55, 2005.

[31] S. Marwan, Y. Shi, I. Menezes, M. Chi, T. Barnes, and T. W. Price. Just a few expert constraints can help: Humanizing data-driven subgoal detection for

novice programming. *International Educational Data Mining Society*, 2021.

[32] P. J. Mensink and K. King. Student access of online feedback is modified by the availability of assessment marks, gender and academic performance. *British Journal of Educational Technology*, 51(1):10–22, 2020.

[33] R. Nagata, M. Vilenius, and E. Whittaker. Correcting preposition errors in learner english using error case frames and feedback messages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–764, 2014.

[34] S. Narciss, S. Sosnovsky, L. Schnaubert, E. Andrès, A. Eichelmann, G. Goguadze, and E. Melis. Exploring feedback and student characteristics relevant for personalizing feedback strategies. *Computers & Education*, 71:56–76, 2014.

[35] A. M. Olney. Generating response-specific elaborated feedback using long-form neural question answering. In *Proceedings of the Eighth ACM Conference on Learning@ Scale*, pages 27–36, 2021.

[36] J. W. Orr and N. Russell. Automatic assessment of the design quality of python programs with personalized feedback. *arXiv preprint arXiv:2106.01399*, 2021.

[37] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[38] A. Poulos and M. J. Mahony. Effectiveness of feedback: The students' perspective. *Assessment & Evaluation in Higher Education*, 33(2):143–154, 2008.

[39] P. Race. Using feedback to help students to learn. *The Higher Education Academy*, 2001.

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[41] J. Roschelle, J. Lester, and J. Fusco. Ai and the future of learning: Expert panel report. *Digital Promise*, 2020.

[42] R. Y. Rubinstein and D. P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2004.

[43] T. Sellam, D. Das, and A. P. Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

[44] S. Subramanian, R. Li, J. Pilault, and C. Pal. On extractive and abstractive neural document summarization with transformer language models. *arXiv preprint arXiv:1909.03186*, 2019.

[45] 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*, 30, 2017.

[46] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[47] W. Wang, G. Fraser, T. Barnes, C. Martens, and T. Price. Execution-trace-based feature engineering to enable formative feedback on visual, interactive programs. *feedback*, 32(1):2, 2021.

[48] N. E. Winstone and D. Boud. The need to disentangle assessment and feedback in higher education. *Studies in Higher Education*, pages 1–12, 2020.

[49] B. Woods, D. Adamson, S. Miel, and E. Mayfield. Formative essay feedback using predictive scoring models. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2071–2080, 2017.

[50] Y. Xiao, G. Zingle, Q. Jia, H. R. Shah, Y. Zhang, T. Li, M. Karovaliya, W. Zhao, Y. Song, J. Ji, et al. Detecting problem statements in peer assessments. *arXiv preprint arXiv:2006.04532*, 2020.

[51] S. Young. Student views of effective online teaching in higher education. *The American Journal of Distance Education*, 20(2):65–77, 2006.

[52] W. Yuan, P. Liu, and G. Neubig. Can we automate scientific reviewing?, 2021.

[53] W. Yuan, G. Neubig, and P. Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34, 2021.

[54] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big bird: Transformers for longer sequences, 2021.

[55] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[56] R. Zhi, S. Marwan, Y. Dong, N. Lytle, T. W. Price, and T. Barnes. Toward data-driven example feedback for novice programming. *International Educational Data Mining Society*, 2019.

[57] G. Zingle, B. Radhakrishnan, Y. Xiao, E. Gehringer, Z. Xiao, F. Pramudianto, G. Khurana, and A. Arnav. Detecting suggestions in peer assessments. *International Educational Data Mining Society*, 2019.