Automated Feedback Generation for Student Project Reports: A Data-Driven Approach

Qinjin Jia
North Carolina State University
qjia3@ncsu.edu

Yunkai Xiao
North Carolina State University
yxiao28@ncsu.edu

Chengyuan Liu
North Carolina State University
cliu32@ncsu.edu

Edward Gehringer
North Carolina State University
efg@ncsu.edu

Mitchell Young
North Carolina State University
mgyoung@ncsu.edu

Jialin Cui
North Carolina State University
jcui9@ncsu.edu

Parvez Rashid
North Carolina State University
mrashid4@ncsu.edu

Instant feedback plays a vital role in promoting academic achievement and student success. In practice, however, delivering timely feedback to students can be challenging for instructors for a variety of reasons (e.g., limited teaching resources). In many cases, feedback arrives too late for learners to act on the advice and reinforce their understanding. To this end, researchers have designed various automated feedback systems in different domains, including novice programming, short-essay writing, and open-ended questions. To the best of our knowledge, no previous study has investigated automated feedback generation for a more complex form of student work - student project reports. In this work, we present a novel data-driven system, dubbed *Insta-Reviewer*, for automatically generating instant feedback on student project reports. In addition to automatic metrics such as ROUGE scores and BERTScore, we propose a five-dimension 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 quality. Our work demonstrates the feasibility of automatic feedback generation for students' project reports while highlighting several prominent challenges for future research.[1]

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

[1]We make all code publicly available: https://github.com/qinjinjia/JEDM22_Automated_Feedback_Generation

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 (Alharbi 2017, Hattie and Timperley 2007, Kusairi 2020). 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 urge them to attain learning goals (Kulik and Kulik 1988, Young 2006, Evans et al. 2014). However, owing to various constraints (e.g., staff availability), feedback often comes too late for students to enact the advice and benefit their learning (Winstone and Boud 2022, Poulos and Mahony 2008, Mensink and King 2020, Kusairi 2020). 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 (Poulos and Mahony, 2008). Thus, tardy feedback can inadvertently position learners as passive recipients of feedback information and limit their engagement with feedback and learning (Carless et al. 2011, Poulos and Mahony 2008).

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 (Marwan et al. 2021, Zhi et al. 2019), short-essay writing (Woods et al. 2017, Botarleanu et al. 2018), and open-ended short answers (Malik et al. 2021, Basu et al. 2013). For instance, Marwan et al. (2021) implemented a hybrid method to deliver instant feedback for block-based novice programming. These and plenty of other impressive studies (e.g., Ariely et al. 2020, Mallavarapu 2020, Lu and Cutumisu 2021, Orr and Russell 2021, Wang et al. 2021) have designed multifarious feedback systems to facilitate student learning and have shown promising results across domains. It can be argued that automatic feedback systems will be integral parts of the future AI-powered educational ecosystem (Roschelle et al., 2020).

It is also worth noting that feedback-generation systems are relevant to, but different from, intelligent tutoring systems (ITSs) (e.g., Mitrovic 2003, Koedinger et al. 2006, Mitrovic 2012), although both provide feedback. Briefly, ITSs are computer systems that are designed to help students master skills by modeling domain knowledge, continuously tracking student progress, and providing customized feedback or hints (Mitrovic 2010, Weitekamp et al. 2020). ITSs provide feedback based primarily on information (e.g., knowledge deficiencies in students) from the domain and student models. Feedback-generation systems, on the other hand, aim to provide textual feedback by modeling the relations between student artifacts and feedback.

While many feedback-generation systems have been proposed, to our knowledge, no attempt has been made to evaluate the feasibility of automatic feedback generation for a more complex form of student work - students' project reports. It is well-known that course projects are an essential part of many university curricula, especially STEM[2] courses (Borowczak et al. 2015, Gehringer 2020). These projects can help students reinforce their theoretical knowledge and develop a host of skills that are increasingly important in the professional world (Mannix and Neale 2005, Gehringer 2020). However, delivering immediate feedback on project reports is often infeasible for instructors due to multifarious reasons. We summarize the reasons why such an automated feedback system for students' project reports is significant as follows:

[2]The STEM field includes science, technology, engineering, and mathematics. They are in high demand across a variety of industries and are significant factors in producing economic prosperity (Tytler, 2020).

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 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 massive open online courses (MOOCs).

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 (Feigenblat et al., 2017), 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 (Lewis et al., 2020) 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 (Lin, 2004), a model-based metric BERTScore (Zhang et al., 2020), 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 student reports?
RQ2: What are the problems of system-generated reviews? What are they not good at?
RQ3: How does the automated feedback system perform in different small-data settings?
RQ4: 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 quality. More specifically, the feedback generated by our system is fluent, coherent, positive, and adept at identifying problems in student reports. However, it may include some non-factual or ambiguous statements in generated feedback. In conclusion, 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.

This manuscript is an extend version of our paper "Insta-Reviewer: A Data-Driven Approach for Generating Instant Feedback on Students' Project Reports," (Jia et al., 2022) which was presented at the 15^{th} International Conference on Educational Data Mining. We have expanded our

experimental results and discussion section to provide more detailed results and more extensive discussions. In addition, we have added a paragraph in the introduction section to elaborate on the discrepancy between the intelligent tutoring systems and our feedback-generation system.

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. Section 5 describes our experimental setup and explains the evaluation metrics, including a new human-evaluation metric. Section 6 through Section 10 present and discuss our experimental results. Section 11 concludes the paper, mentions the limitations of our work, and provides some discussion about future research.

2. Literature Review

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 (Hattie and Timperley, 2007). Previous research has been devoted to designing a variety of feedback systems that provide feedback on various forms of student work, such as 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.

Early studies typically use expert-driven (also called rule-based) methods, which relies on a set of pre-defined rules that encode expert knowledge and experience to provide feedback. For example, Nagata et al. (2014) introduced an expert-driven approach for diagnosing preposition errors and producing feedback for English writing. Despite the fact that such methods often yield accurate feedback and are not data-hungry, they have two significant limitations. First, they are not suitable for dealing with complex, open-ended problems because creating and maintaining a vast collection of expert-designed rules for such problems is nearly impossible. Second, expert-driven methods necessitate extensive human expert effort in constructing feedback rules, and it is hard to generalize them to new tasks and domains (Marwan et al., 2021).

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 implicitly learning the mappings (i.e., rules) from student work to expert feedback by deep-learning algorithms (Deeva et al., 2021). For example, Lu and Cutumisu (2021) implemented several models, including CNN, CNN+LSTM, and CNN+Bi-LSTM, for generating textual feedback on students' essays. However, traditional time-series models are incapable of capturing long-span dependencies in lengthy documents and rely on enormous amounts of training data to learn underlying patterns (Tang et al., 2018).

More recent work has begun to use large-scale pre-trained language models, such as BERT (Devlin et al., 2019), BART (Lewis et al., 2020), and GPT-2 (Radford et al., 2019), for generating feedback on open-ended student work. These language models use the attention mechanism (Vaswani et al., 2017) to learn long-span dependencies, and are pre-trained on large generic corpora in an unsupervised manner (i.e., transfer learning) to reduce the need for labeled data. For instance, Olney (2021) attempted to generate elaborated feedback for student responses using the ELI5 model (Fan et al., 2019) and achieved promising results. In this paper, we also utilize such a pre-trained language model to exploit its ability to capture long-range dependencies in student reports and to take advantage of pre-training.

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 (Celikyilmaz et al., 2020).

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 (Celikyilmaz et al., 2020). 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 (Papineni et al., 2002) and ROUGE (Lin, 2004).

BLEU (the Bilingual Evaluation Understudy) is a precision-based content-overlap metric proposed by Papineni et al. (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 (Callison-Burch et al., 2006).

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the other most commonly used content-overlap metric introduced by Lin (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 (Celikyilmaz et al., 2020). Given the excellent performance of BERT (Devlin et al., 2019) 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 (Zhang et al. 2020, Yuan et al. 2021, Sellam et al. 2020). One of the most widely used metrics is BERTScore (Zhang et al., 2020).

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 (2013) found that a positive tone and suggestions for improvement are key features of good reviews. Jia et al. (2021) mentioned that providing suggestions, mentioning problems, and using a positive tone, are main characteristics of effective feedback. Celikyilmaz et al. (2020) 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 5.1.3.

2.3. Ethical Considerations in Feedback Generation

To date, ethical transgressions in feedback generation have received scant attention in the research literature, despite the fact that text-generation methods potentially raise certain ethical concerns. Celikyilmaz et al. (2020) and Gehman et al. (2020) pointed out that generating improper or offensive language is an ethical issue that may appear in text generation. Bender et al. (2021) and Li et al. (2022) also mentioned that using pre-trained language models may cause issues, such as producing personally identifiable information, because corpora used for pre-training are pulled from the internet without careful filtering. Utilization of such uncurated data may lead to models replicating harmful language or personal information that emerged during the pre-training process (Malmi et al., 2022).

Our feedback-generation systems may be susceptible to the aforementioned issues, as we also employ pre-trained language models in our approaches. Although we have mitigated the concerns by fine-tuning the models using filtered domain-specific data, it is still vital to investigate potential ethical violations with our feedback-generation systems. However, systematic methods to assess how effectively a system can avoid generating inappropriate language are still lacking (Celikyilmaz et al. 2020, Illia et al. 2022). For these reasons and others, this work monitors the ethics of feedback-generation models for student project reports by manually inspecting all system-generated feedback. In addition, we strongly encourage researchers in the community to pay attention and make efforts to explore potential ethical violations in text generation.

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.

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.

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[3][4][5]. Three randomly sampled[6] instructor reviews from our dataset are displayed in Table 1.

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

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

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

[6]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[7] 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, including the number of words and tokens of our data. Note that since the pre-trained language model BART (detailed in Section 4) has a maximum input length limit of 1024 tokens[8], 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, which is the input length limit of the BART model. Thus, we summarized the reports before feeding them into BART. 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.

Table 2: Statistics on the Dataset. Note that 75.8% of the project reports comprise more than 1024 tokens. 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**
			5th	25th	50th	75th	95th	
Reports	# of words	1193	405	734	1060	1499	2374	6512
	# of tokens	1643	583	1040	1489	2025	3272	8422
Summarized Reports	# of words	704	395	649	742	803	865	908
	# of tokens	951	580	1004	1021	1023	1024	1024
Reference Reviews	# of words	55	13	33	48	72	114	258
	# of tokens	71	19	43	61	90	147	335

[7]The Family Educational Rights and Privacy Act (FERPA) (20 U.S.C. § 1232g; 34 CFR Part 99) is a Federal law that protects the privacy of student education records.

[8]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.

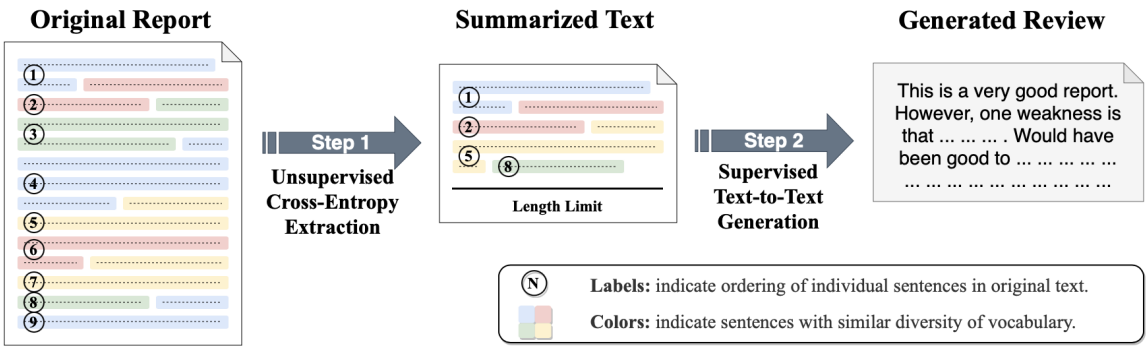

Figure 1: Operation of our *Insta-Reviewer* System. The system uses a select-then-generate paradigm (Pilault et al. 2020, Gehrmann et al. 2018, Yuan et al. 2022). The first step is to extract salient sentences (within the length limit) from students' project reports. The second step is to utilize a supervised 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.

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 (Lewis et al., 2020).

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[9] (Lewis et al., 2020). 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 10.1). Thus, we adopt a select-then-generate method (Gehrmann et al. 2018, Pilault et al. 2020, Yuan et al. 2022) 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*.

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 pre-trained 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 (Feigenblat et al., 2017).

The cross-entropy method is an unsupervised technique that treats sentence-level extractive summarization as a combinatorial optimization problem (Rubinstein and Kroese, 2004). 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

[9]The length is limited to 1024 since the BART authors (Lewis et al., 2020) chose this number and pre-trained the model with this limit.

summary S, as shown below,

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

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

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 $\textsc{Len(S)}$ 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 $\textsc{Tokens(S)}$, 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.

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

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, $X_1, X_2, \ldots, X_N$ according to the probability vector p_{t-1}. The sentences selected from each of these vectors define our N sample summaries $S_1, S_2, \ldots, S_N$. 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: $\beta = 5/1024$, $N = 1000$, $\rho = .05$, $\alpha = .7$. 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 (Lewis et al., 2020), 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).

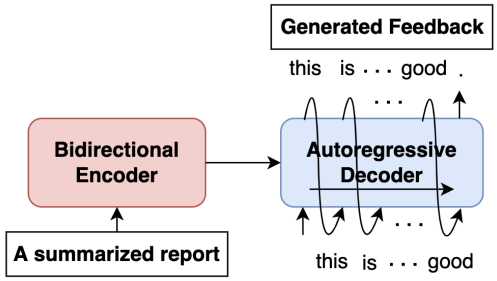

Figure 2: Illustration of an encoder-decoder architecture. The encoder can convert an input sequence of text (e.g., a summarized report) into a rich numerical representation, and then the decoder generates the output sequence (e.g., feedback) by iteratively predicting the most probable next word.

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 (Erhan et al., 2010). In this work, we use the pre-trained checkpoint "facebook/bart-large-cnn[10]" to initialize all parameters of BART and then fine-tune BART 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) (Vijayakumar et al., 2017) 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. Similarly to the beam search, 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. However, in order to improve the diversity in the outputs, all beams in DBS 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 used in the ablation experiments (mentioned in Section 10.2).

5. Evaluation and Experimental Setting

In this section, we introduce our evaluation metrics and describe experimental settings. Section 5.1 presents a set of metrics, which includes a human evaluation metric, for evaluating generated textual feedback. Then, Section 5.2 introduces training details and our hardware setup.

5.1. Evaluation Metrics

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 Insta-Reviewer 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 (Celikyilmaz et al., 2020). 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.

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

5.1.1. 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[11] (Lin 2004, Ganesan 2006). Porter stemming is enabled to remove plurals and word suffixes (e.g., "ing", "ion", "ment").

5.1.2. 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[12] (Zhang et al., 2020).

5.1.3. Human Evaluation

After reviewing relevant manuscripts (e.g., Caligiuri and Thomas 2013, Jia et al. 2021, Xiao et al. 2020, Zingle et al. 2019) 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.

(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

[11] https://github.com/google-research/google-research/tree/master/rouge
[12] https://github.com/Tiiiger/bert_score

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.

5.2. Experimental Setup

5.2.1. 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 (Loshchilov and Hutter, 2019) 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.

5.2.2. Hardware Setup

The BART models are trained on an NVIDIA RTX6000 GPU (24GB). The BigBirdPegasus model (mentioned in Section 10.2) 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.

6. Study 1: Performance of Feedback Generation

The goal of our first experiment is to find out if Insta-Reviewer is effective in generating readable and helpful feedback for students' project reports (RQ1). We tested our feedback-generation system on the test set (n=50) and compared its performance with human-written feedback. The evaluation results for Insta-Reviewer are shown in Table 3.

6.1. Experimental Results from Study 1

In the first row of Table 3, we show human evaluation scores for the feedback provided by instructors. In the second row, we present the evaluation results for Insta-Reviewer, including the ROUGE scores, BERTScore, and the human evaluation scores. As mentioned in Section 5.1, expert feedback is used as ground-truth labels in the calculation of the ROUGE scores and BERTScores for system-generated feedback. Thus, there are no ROUGE scores and BERTScore for expert feedback, as the expert reviews are considered to be ground-truth feedback.

According to Table 3, the ROUGE-1 (R1), ROUGE-2 (R2), ROUGE-Lsum (RLsum), and BERTScore (BRTS) for our Insta-Reviewer ("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, in n-grams). More intuitively, the generated feedback for some actual student reports is shown in Table 4, and their corresponding evaluation scores are shown in Table 5.

For example, the scores (R1=27.52, R2=1.87, RLsum=20.18, BRTS=59.17) of the generated feedback 1 as shown in Table 4 are similar to the average ROUGE scores and BERTScore (R1=28.54, R2=6.39, RLSum=18.21, BRTS=59.18) of Insta-Reviewer. By comparing the text

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. All human-evaluation scores are reported in percentages. BRTS=BERTScore, READ=Readability, SUGG=Suggestions, PROB=Problems, TONE=Positive Tone, and FACT=Factuality.

	Method	ROUGE			BRTS	Human Evaluation				
		R1	R2	RLSum		READ	SUGG	PROB	TONE	FACT
	1. Expert Feedback	-	-	-	-	**100.0**	**82.0**	90.0	93.0	**100.0**
Insta-Reviewer	2. CE + BART	**28.54**	**6.39**	**18.21**	**59.18**	94.0	66.0	**96.0**	**95.0**	84.8

of the "generated feedback 1" and the "expert feedback 1", we can find that the semantics of these two reviews are essentially consistent. More concretely, both the "generated feedback 1" and the "expert feedback 1" mention that the report is highly readable and explains the changes in detail. They also both identify the problem that the report does not provide a test plan, which is very useful for verifying changes. However, the "expert feedback 1" mentions that the pictures in the report are taken with a phone camera, which is not mentioned in the generated feedback.

For human evaluation scores, 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., the report is missing a test plan) rather than project-specific issues (e.g., "xxx files should be described in detail"). Additionally, the expert feedback can outperform the generated feedback with gaps of 6%, 16%, 15.2% for "Readability," "Suggestions," and "Factuality," respectively. The results imply that although our system is not as good as experts at providing suggestions and may produce some non-factual or ambiguous statements, it is good at generating fluent and positive feedback that mentions problems that need to be addressed.

6.2. Discussion of Study 1

In the first study, we evaluated our feedback-generation system, Insta-Reviewer, by calculating its ROUGE scores and BERTScore, and comparing its human evaluation scores with those of expert feedback. Overall, the results demonstrate that the system-generated feedback is generally semantically consistent (BERTScore=59.18) with the expert feedback. However, the generated feedback may not contain some of the points mentioned in the expert feedback, but it may also identify some issues that are not mentioned in the expert reviews. The human evaluation scores indicate that our feedback-generation system is effective in producing feedback that is helpful to students for their project reports. The generated feedback can outperform human experts in terms of "Problems" and "Positive Tone," but it does not do well in generating "Suggestions" and may occasionally produce non-factual or ambiguous statements.

More specifically, we observed from Table 3 that there is a relatively large gap (16.0%) between the system-generated feedback and the expert feedback on the "Suggestions" dimension. We speculate that this is due to the fact that "suggestion" statements in reviews are usually quite diverse, and it is relatively challenging to learn latent relations between student project reports and the corresponding suggestions. We also cannot exclude the possibility that many "suggestion" statements focus on particular sentences or paragraphs of the report, but these sentences may have been truncated in the summarization step of the select-then-generate paradigm. Fu-

ture research could carefully investigate the reasons for the system's inferior performance on the "Suggestions" dimension and target improvements to our current feedback-generation method.

In addition, the human evaluation metrics proposed in the paper are not perfect and have potential limitations. For example, the evaluation dimension "Problems" does not take into account the quality of the problems mentioned in the feedback (i.e., how helpful the mentioned problems were to the students). Empirically, project-specific problems may motivate students to revise their projects better than generic problems. However, our evaluation dimension "Problems" do not take this discrepancy into account and may therefore overestimate the quality of the generated feedback. While we believe these limitations of the human evaluation metrics do not impact the primary outcome of the study, future work could seek to improve the human evaluation metrics to make it possible to evaluate generated feedback more accurately.

In conclusion, the first study suggests that building automated feedback systems is feasible not only for short-answer questions and programming problems, but also for more complicated and open-ended student work such as project reports. Nevertheless, these more complex forms of student work require data-driven approaches to learn the underlying patterns between student work and feedback, as the patterns are typically too intricate to be explicitly created and maintained by expert-driven methods (i.e., rule-based methods). However, data-driven approaches also have their drawbacks: they are data hungry (i.e., require sufficient data to effectively capture the latent patterns between student work and feedback), and their generation is not always controllable (e.g., may produce some non-factual statements). Future work could therefore attempt to alleviate the drawbacks by combining data-driven and expert-driven approaches.

7. Study 2: Issues in system-generated reviews

In the second experiment, we explored the potential deficiencies of system-generated feedback in more detail (RQ2). The first study evaluated the performance of our feedback-generation system from an overall perspective, but it might have overlooked some potential detailed flaws in the generated feedback. To this end, we manually examined all system-generated feedback in the test set (n=50). Although the vast majority of the feedback was fluent, positive, and factually correct, we found four potential imperfections in the system-generated feedback upon inspection. Table 4 shows three randomly sampled system-generated reviews, and Table 5 displays their corresponding ROUGE scores, BERTScore, and human evaluation scores.

7.1. Experimental results from Study 2

In Table 4, we show system-generated reviews and expert reviews for three project reports from real classes. For the first report, the "expert feedback 1" and the "generated feedback 1" contain some overlapping words and similar semantics. They both mentioned that the report documented the changes well and pointed out that the report lacked a test plan. However, since our feedback-generation system merely utilized the text in the report as input, the issue that "the pictures in reports were taken with a phone camera" was not mentioned in the generated feedback. For the "generated feedback 1," the ROUGE scores (R1=27.52, R2=1.87, RLsum=20.18) and BERTScore (BRTS=59.17) are roughly consistent with our expectation. In addition, the "generated feedback 1" scored 100 on all human evaluation dimensions, as it was coherent, used positive tone, mentioned problems, provided suggestions, and all statements are factual.

For the second report, the "expert feedback 2" and the "generated feedback 2" both said that

the changes to the code should be described, as shown in Table 4. The generated feedback also detected that the report did not describe the test plan. However, the "generated feedback 2" 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." 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 are testing." In addition, the "generated feedback 2" was smooth and positive, but did not provide any valid suggestion. Thus, it scored 0 and 83.33 on "Suggestions" and "Factuality," respectively, while scoring 100 for the other three dimensions.

For the third report, there is a salient semantic difference between the "expert feedback 3" and the "generated feedback 3." The expert review found the report to be well-written but did not mention design patterns, and the review also provided a project-specific suggestion - "Gemfiles should be downgraded." However, the "generated feedback 3" did not contain any project-specific statements, but it simply encouraged the reviewee to include more explanation of the changes. In addition, the "generated feedback 3" also contained a non-factual statement - "the changes are described well." and a vague indicator pronoun ("they"). From the third row of Table 5, we can also find that there is a huge gap between the evaluation scores (R1=22.22, R2=0.0, RLsum=13.89, BRTS=52.92) of the "generated feedback 3" and the average scores (R1=28.54, R2=6.39, RLsum=18.21, BRTS=59.18) of all system-generated feedback.

7.2. Discussion of Study 2

In the second experiment, we manually inspected all system-generated feedback and found four potential deficiencies: non-factual statements, frequent pieces of text, lack of project-specific "problem" or "suggestion" statements, and inability to provide feedback on images in project reports. These findings highlight the salient challenges for further research on this topic, and we encourage researchers to explore solutions to the issues to advance our feedback-generation approach. In the following paragraphs, we will discuss these four issues in the system-generated feedback.

First, the automated feedback system may occasionally ($\approx$15.2% of all statements) generate some non-factual or ambiguous statements in the feedback. These non-factual or vague statements contained in the system-generated feedback may mislead or confuse students and have a negative impact on students' learning. Completely eliminating these inaccurate sentences is almost impossible for a data-driven approach. However, for such data-driven feedback-generation models, increasing the amount of training data and improving the data quality can typically mitigate this issue. An alternative solution is to train an additional "filter" network to filter out non-factual statements before delivering them to students.

Second, we also found that a few pieces of text frequently appear in different generated feedback. For instance, all three generated reviews shown in Table 4 start with the sentence "the writeup is very/quite readable." After counting, we found that this phrase was contained in 66% of the system-generated feedback. 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 in training. This repetition problem is not necessarily a drawback of the system, but it suggests that high-frequency text pieces in the training dataset can influence the generation. Future research could explore whether such frequent text snippets can negatively affect learners, and how to eliminate the impact.

Third, the system-generated feedback lacks project-specific "problem" or "suggestion" state-

Table 4: Sample expert and generated feedback. Evaluation scores for the generated feedback are shown in Table 5. 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"

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

	ROUGE			BERTScore	Human Evaluation				
	R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT
Test set average (n=50)	28.54	6.39	18.21	59.18	94.0	66.0	96.0	95.0	84.8
Generated Feedback 1	27.52	1.87	20.18	59.17	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
Generated Feedback 3	22.22	0.00	13.89	52.92	80.00	0.00	100.0	100.0	80.00

ments. We note that our feedback-generation system tends to identify common problems that frequently appear in different reports (e.g., the test plan is missing) and produce some generic suggestion (e.g., add a explanation of the changes). However, such broad feedback may not be as valuable to students as project-specific feedback (e.g., "Gemfiles should be downgraded"). In our experiments, we observed that there might be a trade-off between the specificity and the factuality of the feedback generation. Increasing the beam size of the decoding algorithm (i.e., the diverse beam search algorithm introduced in Section 4.4.2) would improve the factuality and decrease the specificity of the generated feedback, and vice versa. Extensive experiments should be conducted in the future to investigate how the beam size affects the generation.

Lastly, our approach is unable to provide feedback on pictures in project reports. One of the limitations of our approach is that our system can merely provide feedback for textual information in reports, as our feedback-generation model cannot incorporate image information into the input. This limitation of our system results in students not being able to know if there is a issue with the pictures in their reports and how to improve them. One potential solution is to train a separate system to provide feedback on the images in student reports. Another promising approach is to upgrade our current model to a multi-modal model, so that the system can provide feedback on all text, images, and even other artifacts such as code.

8. Study 3: Performance in small-data settings

In the third study, we would like to investigate the performance of our automated feedback system in different small-data settings (RQ3). As we mentioned, our dataset contains 484 project reports and associated reviews from twelve semesters. However, collecting this amount of data can be a challenge for many courses. Therefore, in order to help other researchers or instructors who intend to apply our approach to their datasets, we evaluated the effectiveness of our approach in different small-data settings (≤ 100 training samples). More specifically, we calculated and compared the ROUGE scores and BERTScore of system-generated feedback when we trained Insta-Reviewer with 50, 100, and all available (i.e., 434) training data samples, respectively. The evaluation results of Insta-Reviewer in the small-data settings are shown in Table 6.

8.1. Experimental results from Study 3

In the first, the second, and the third row of Table 6, we show ROUGE scores and BERTScore (with a 95% confidence interval) when training Insta-Reviewer with all available training data (i.e., 434 samples), 50 samples, and 100 samples, respectively. According to Table 6, the

ROUGE-1, ROUGE-2, ROUGE-Lsum, and BERTScore of Insta-Reviewer when training with 434 data samples are 28.54±2.98, 6.39±1.56, 18.21±1.83, and 59.18±1.54, respectively. When training with 50 data samples, the ROUGE scores and BERTScore decreased by 4.25, 1.4, 0.79, and 3.06, respectively. When we trained Insta-Reviewer with 100 data samples, the ROUGE scores and BERTScore dropped by 2.79, 0.83, 0.35, and 1.82, respectively. These scores imply that when trained with 50 or 100 samples, the feedback generated by Insta-Reviewer still has a relatively high similarity to the expert feedback. Thus, our method can potentially be applied to other tasks that have limited data.

Table 6: ROUGE F_1 (with porter stemming) and BERTScore on the test set (n=50) in small-data settings ($\leq$ 100 training samples). For each score, we report a 95% confidence interval (CI). The first column ("Data points") indicates the number of report-feedback pairs used for training.

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

8.2. Discussion of Study 3

In the third study, we evaluated the performance of Insta-Reviewer in different small-data settings (training the model with 50 or 100 data samples). The experimental results demonstrated that our feedback-generation approach could potentially be used in scenarios where the amount of data is limited. However, it is worth noting that there may be many other factors that can impact the performance of the model in small-data conditions. For example, if expert feedback is very long, the modal may need more data to capture the underlying relations between student reports and feedback. In this case, the performance of our approach may degrade rapidly after reducing the amount of training data. Next steps include attempting to reduce the need for quality training data, investigating whether it is feasible to synthesize some pseudo-data for training, and exploring whether a feedback-generation system can be generalized to multiple domains.

9. Study 4: Ethical concerns with the system

The objective of the fourth experiment is to investigate whether the automated feedback system raises any ethical concerns (RQ4). As we mentioned in Section 2.3, there is still a lack of systematic methods to assess how reliably a model can avoid generating inappropriate content. Thus, we evaluated the potential ethical issues by manually examining all generated feedback.

9.1. Experimental results from Study 4

In this work, we consider two main ethical issues related to the system. The first primary concern is whether the system will generate improper or offensive language. Although we have filtered out all inappropriate information from our dataset, this ethical concern may still arise since the BART model is pre-trained on a large-scale corpus crawled from the Internet without fine-grained filtering. Thus, the model may replicate inappropriate content appearing in the pre-training corpus in the generated feedback. The second issue is whether the generated feedback

will contain some private content (e.g., names and email addresses). Although we have carefully removed all private information from our dataset, there may still be omissions that could potentially appear in the generated feedback. Therefore, we manually vetted all generated feedback and failed to find any of the aforementioned ethical transgressions.

9.2. Discussion of Study 4

In the fourth study, we carefully inspected all system-generated feedback to check whether Insta-Reviewer raised any ethical concerns. Inappropriate language in system-generated feedback could hurt some students in unforeseen ways and even raise legal issues. Thus, preventing the model from generating offensive language or private content is crucial to whether the model can be deployed in actual classes. Although we did not observe any ethical issues in the generated feedback, this is not complete proof that our system will never generate inappropriate content. A simple solution would be to have the instructors manually check the generated content before delivering the system-generated feedback to learners. However, this solution would prevent students from receiving instant feedback for their project reports. Future work should design systematic methods to check whether the model can avoid generating inappropriate content.

10. Ablation Experiments

In order to understand the contribution of each step in our feedback-generation approach, we designed two ablation experiments. The goal of the first ablation experiment was to verify our hypothesis that using cross-entropy extraction (CE) to summarize the original students' project reports is better than simply truncating all tokens beyond the length limit of BART. The second ablation experiment aimed to know whether the sparse-attention-based pre-trained language model (PLM) BigBirdPeagsus (Zaheer et al., 2020) can effectively fit the complex underlying mappings from student reports to textual feedback and replace our "CE + BART" approach. The evaluation results for the two ablation experiments are presented in Table 7. 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 results of the first and the second ablation experiments are analyzed and discussed in Section 10.1 and 10.2, respectively.

Table 7: 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. Note that the BERTScore script does not support BigBirdPegasus currently.

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

10.1. Ablation Experiment 1 - Naïve BART:

In the first ablation experiment, we investigate the contribution of the cross-entropy (CE) summarization to the results obtained with the system. 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 for the entire feedback-generation system.

By looking at the ROUGE scores and BERTScore shown in Table 7, 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, respectively. The higher ROUGE scores suggest that the feedback generated by the "CE+BART" method has more word or n-gram overlap with the expert feedback than the feedback generated by the "naïve BART" method. Similarly, the better BERTScore demonstrates that the feedback produced by the "CE+BART" method is also more semantically consistent with the feedback provided by instructors than the "naïve BART" method. Thus, employing the CE step to summarize the reports, rather than simply truncating all tokens that exceed the length limit, allows for better surface word matching and more consistent semantics between the generated feedback and the expert feedback.

Additionally, based on the human evaluation scores shown in Table 7, "CE + BART" significantly ($\geq 8\%$) outperforms the "Naïve BART" method on the "Readability," "Suggestions," and "Factuality" evaluation dimensions. This implies that the feedback generated by "CE+BART" method is less ambiguous, contains more suggestions, and is more accurate. The "CE + BART" method can also achieve higher ($\geq 2\%$) scores on dimensions "Problems" and "Positive tone." This suggests that the "CE+BART" method is more likely to identify issues in student project reports and uses slightly more positive tone than the "Naïve BART" method. Therefore, we can conclude that the "CE + BART" method is more effective than the "naïve BART" method in generating feedback in terms of human evaluation scores.

Overall, the results demonstrate that the CE method can effectively summarize students' reports while retaining critical information that helps the model generate feedback. Thus, the CE summarization step contributes to the performance of the feedback-generation system for student project reports. In addition, based on the human evaluation scores, an interesting observation is that utilizing "CE + BART" instead of "Naïve BART" has greater impacts ($\geq 8\%$) on the "Readability," "Suggestions," and "Factuality" evaluation dimensions. Nevertheless, the gaps for dimensions "Problems" and "Positive tone" are not very significant ($\geq 2\%$). We do not have an explanation for this finding yet, but future work could investigate the reasons for it.

10.2. Ablation Experiment 2 - BigBirdPegasus:

In the second ablation experiment, we investigate whether the "CE + BART" method can be replaced by the recently proposed sparse-attention-based PLM BigBirdPegsus (Zaheer et al., 2020), 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 capture complex latent relations between the project reports and feedback. Similar to BART, BigBirdPegasus also consists of pre-training and fine-tuning steps. In this experiment, we use the pre-trained checkpoint "google/bigbird-pegasus-large-arxiv[13]" to initialize all BigbirdPegasus parameters and then fine-tune the model on our data. If BigBirdPegasus can outperform our "CE + BART" method, it will demonstrate that levering the select-then-generate paradigm is not necessary. Otherwise, the results will suggest that our "CE + BART" method is superior to BigBirdPegasus for generating feedback.

As shown in Table 7, the results indicate that "CE + BART" substantially outperforms the BigBirdPeagsus model on all metrics. For the automatic metric ROUGE scores, our "CE + BART" method can outperform BigBirdPegasus, with gaps of 12.62, 2.35, and 4.76 for R1, R2, and RLsum, respectively. These scores imply that the feedback generated by BigBirdPegasus essentially does not have much surface word or n-gram overlap with the expert feedback. For the human evaluation scores, the "CE + BART" method can substantially outperform BigBirdPegasus with gaps of 38%, 64%, 84%, 37%, and 61.4% for "Readability," "Suggestions," "Problems," "Positive Tone," and "Factuality," respectively. This suggests that the feedback generated by BigBirdPegasus is not as fluent and coherent as the "CE + BART" method, and BigBirdPegasus rarely generates valid suggestions or problem statements. Thus, we can conclude that the "CE + BART" method is more effective in generating feedback than BigBirdPegasus.

Overall, 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. However, we cannot exclude the possibility that the BigBirdPeagsus model is more powerful than our "CE + BART" method, if sufficient training data is available. Nevertheless, we cannot verify or falsify this hypothesis, since we only have 484 student reports and corresponding expert reviews. Future work could conduct more extensive ablation experiments to compare the "CE + BART" method and BigBirdPeagsus on feedback generation.

11. Conclusion, Limitations, and Future Research

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.

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

11.1. LIMITATIONS

We acknowledge the limitations of our research that must be further investigated. In summary, there are four major limitations to this study.

Firstly, due the the lack of public datasets, our experiments were performed on a single corpus from a graduate-level computer science class. Thus, it is unclear whether our approach and findings can be generalized to other corpora or domains. Many unforeseen factors may affect the quality of system-generated feedback. For example, for those datasets with longer student reports, our CE summarization step may result in too many salient sentences being abridged.

Second, 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.

Third, 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.

Finally, 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 text that violates privacy or is offensive to readers. 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.

11.2. FUTURE RESEARCH

We have deployed the feedback-generation system in actual classes. However, instead of delivering system-generated feedback to students directly, we use a human-in-the-loop approach to manually filter out all non-factual statements before returning feedback. Based on the perceptions of the course instructors who manually vetted the system-generated feedback, the generated feedback on student project reports is positive, beneficial, and can considerably reduce the grading burden. However, the main obstacle currently preventing a fully automated feedback-generation system is those system-generated non-factual statements. Thus, an important direction for future work is to investigate how to avoid generating non-factual statements in feedback.

The 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 non-factual statements (or let students know the certainty of each statement) before delivering the feedback to students. Dušek and Kasner (2020) have proposed an entailment-based model to evaluate the correctness of 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 some form of combination of expert-driven and data-driven approaches, to unify the benefits of both. For example, Marwan et al. (2021) have proposed a combination of these two approaches to provide hints for novice programming. Future research could adopt the method to feedback generation.

REFERENCES

ALHARBI, W. 2017. E-feedback as a scaffolding teaching strategy in the online language classroom. *Journal of Educational Technology Systems 46,* 2, 239–251.

ARIELY, M., NAZARETSKY, T., AND ALEXANDRON, G. 2020. Fisrt steps towards NLP-based formative feeadback to imporove scientific writing in Hebrew. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, A. N. Rafferty, C. R. Jacob Whitehill, and V. Cavalli-Sforza, Eds. International Educational Data Mining Society, Montreal, Canada, 565–568.

BASU, S., JACOBS, C., AND VANDERWENDE, L. 2013. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics 1*, 391–402.

BENDER, E. M., GEBRU, T., MCMILLAN-MAJOR, A., AND SHMITCHELL, S. 2021. On the dangers of stochastic parrots: Can language models be too big? In *FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. Vol. 1. Association for Computing Machinery, New York, NY, 610–623.

BOROWCZAK, M. ET AL. 2015. Communication in stem education: A non-intrusive method for assessment & k20 educator feedback. *Problems of Education in the 21st Century 65,* 1, 18–27.

BOTARLEANU, R.-M., DASCALU, M., SIRBU, M.-D., CROSSLEY, S. A., AND TRAUSAN-MATU, S. 2018. Readme–generating personalized feedback for essay writing using the readerbench framework. In *Conference on Smart Learning Ecosystems and Regional Development*, H. Knoche, E. Popescu, and A. Cartelli, Eds. SLERD '18. Springer, 133–145.

CALIGIURI, P. AND THOMAS, D. C. 2013. From the editors: How to write a high-quality review. *Journal of International Business Studies 44,* 6, 547–553.

CALLISON-BURCH, C., OSBORNE, M., AND KOEHN, P. 2006. Re-evaluating the role of bleu in machine translation research. In *11th conference of the european chapter of the association for computational linguistics*, D. McCarthy and S. Wintner, Eds. Association for Computational Linguistics, Trento, Italy, 249–256.

CARLESS, D., SALTER, D., YANG, M., AND LAM, J. 2011. Developing sustainable feedback practices. *Studies in higher education 36,* 4, 395–407.

CELIKYILMAZ, A., CLARK, E., AND GAO, J. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

DEEVA, G., BOGDANOVA, D., SERRAL, E., SNOECK, M., AND DE WEERDT, J. 2021. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education 162*, 104094.

DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

DUŠEK, O. AND KASNER, Z. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, B. Davis, Y. Graham, J. Kelleher, and Y. Sripada, Eds. Association for Computational Linguistics, Dublin, Ireland, 131–137.

ERHAN, D., COURVILLE, A., BENGIO, Y., AND VINCENT, P. 2010. Why does unsupervised pretraining help deep learning? In *Proceedings of the thirteenth international conference on artificial*

intelligence and statistics, Y. W. Teh and D. M. Titterington, Eds. JMLR Workshop and Conference Proceedings, Sardinia, Italy, 201–208.

EVANS, D. J., ZEUN, P., AND STANIER, R. A. 2014. Motivating student learning using a formative assessment journey. *Journal of anatomy 224,* 3, 296–303.

FAN, A., JERNITE, Y., PEREZ, E., GRANGIER, D., WESTON, J., AND AULI, M. 2019. Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, Florence, Italy, 3558–3567.

FEIGENBLAT, G., ROITMAN, H., BONI, O., AND KONOPNICKI, D. 2017. 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*, N. Kando, T. Sakai, and H. Joho, Eds. SIGIR '17. Association for Computing Machinery, New York, NY, USA, 961–964.

GANESAN, K. 2006. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *Computational Linguistics 1,* 1.

GEHMAN, S., GURURANGAN, S., SAP, M., CHOI, Y., AND SMITH, N. A. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, online, 3356–3369.

GEHRINGER, E. F. 2020. A course as ecosystem: melding teaching, research, and practice. In *2020 ASEE Virtual Annual Conference Content Access*. online.

GEHRMANN, S., DENG, Y., AND RUSH, A. M. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, Brussels, Belgium, 4098–4109.

HATTIE, J. AND TIMPERLEY, H. 2007. The power of feedback. *Review of educational research 77,* 1, 81–112.

ILLIA, L., COLLEONI, E., AND ZYGLIDOPOULOS, S. 2022. Ethical implications of text generation in the age of artificial intelligence. *Business Ethics, the Environment & Responsibility 32*, 201–210.

JIA, Q., CUI, J., XIAO, Y., LIU, C., RASHID, P., AND GEHRINGER, E. 2021. All-in-one: Multi-task learning bert models for evaluating peer assessments. In *Proceedings of the 14th International Conference on Educational Data Mining*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 474–479.

JIA, Q., YOUNG, M., XIAO, Y., CUI, J., LIU, C., RASHID, P., AND GEHRINGER, E. 2022. InstaReviewer: A Data-Driven Approach for Generating Instant Feedback on Students' Project Reports. In *Proceedings of the 15th International Conference on Educational Data Mining*, A. Mitrovic, N. Bosch, A. I. Cristea, and C. Brown, Eds. Number July in EDM '22. International Educational Data Mining Society, Durham, United Kingdom, 5–16.

KOEDINGER, K. R., CORBETT, A., ET AL. 2006. *Cognitive tutors: Technology bringing learning sciences to the classroom*. Cambridge University Press.

KULIK, J. A. AND KULIK, C.-L. C. 1988. Timing of feedback and verbal learning. *Review of educational research 58,* 1, 79–97.

KUSAIRI, S. 2020. 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.

LEWIS, M., LIU, Y., GOYAL, N., GHAZVININEJAD, M., MOHAMED, A., LEVY, O., STOYANOV, V., AND ZETTLEMOYER, L. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Association for Computational Linguistics, Seattle, Washington, 7871–7880.

LI, J., TANG, T., ZHAO, W. X., NIE, J.-Y., AND WEN, J.-R. 2022. A survey of pretrained language models based text generation. *arXiv preprint arXiv:2201.05273*.

LIN, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, M.-F. Moens and S. Szpakowicz, Eds. Association for Computational Linguistics, Barcelona, Spain, 74–81.

LOSHCHILOV, I. AND HUTTER, F. 2019. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. *arXiv preprint arXiv:1711.05101*.

LU, C. AND CUTUMISU, M. 2021. Integrating deep learning into an automated feedback generation system for automated essay scoring. In *Proceedings of the 14th International Conference on Educational Data Mining*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 573–579.

MALIK, A., WU, M., VASAVADA, V., SONG, J., COOTS, M., MITCHELL, J., GOODMAN, N., AND PIECH, C. 2021. Generative grading: Near human-level accuracy for automated feedback on richly structured problems. In *Proceedings of the 14th International Conference on Educational Data Mining*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 275–286.

MALLAVARAPU, A. 2020. Exploration maps, beyond top scores: Designing formative feedback for open-ended problems. In *Proceedings of the 13th International Conference on Educational Data Mining*. International Educational Data Mining Society, online, 790–795.

MALMI, E., DONG, Y., MALLINSON, J., CHUKLIN, A., ADAMEK, J., MIRYLENKA, D., STAHLBERG, F., KRAUSE, S., KUMAR, S., AND SEVERYN, A. 2022. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*. Association for Computational Linguistics, Seattle, United States, 1–7.

MANNIX, E. AND NEALE, M. A. 2005. What differences make a difference? the promise and reality of diverse teams in organizations. *Psychological science in the public interest 6,* 2, 31–55.

MARWAN, S., SHI, Y., MENEZES, I., CHI, M., BARNES, T., AND PRICE, T. W. 2021. Just a few expert constraints can help: Humanizing data-driven subgoal detection for novice programming. In *Proceedings of the 14th International Conference on Educational Data Mining*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 68–80.

MENSINK, P. J. AND KING, K. 2020. 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.

MITROVIC, A. 2003. An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education 13,* 2-4, 173–197.

MITROVIC, A. 2010. Modeling domains and students with constraint-based modeling. In *Advances in Intelligent Tutoring Systems*, R. Nkambou, J. Bourdeau, and R. Mizoguchi, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 63–80.

MITROVIC, A. 2012. Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User modeling and user-adapted interaction 22,* 1, 39–72.

NAGATA, R., VILENIUS, M., AND WHITTAKER, E. 2014. 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)*, H. W. Kristina Toutanova, Ed. Association for Computational Linguistics, Baltimore, Maryland, 754–764.

OLNEY, A. M. 2021. Generating response-specific elaborated feedback using long-form neural question answering. In *Proceedings of the Eighth ACM Conference on Learning@ Scale*, M. Pérez-Sanagustín, A. Ogan, and M. Specht, Eds. Association for Computing Machinery, New York, NY, 27–36.

ORR, J. W. AND RUSSELL, N. 2021. Automatic assessment of the design quality of python programs with personalized feedback. In *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 495–501.

PAPINENI, K., ROUKOS, S., WARD, T., AND ZHU, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, P. Isabelle, Ed. The association for computational linguistics, Philadelphia, Pennsylvania, USA, 311–318.

PILAULT, J., LI, R., SUBRAMANIAN, S., AND PAL, C. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, Online, 9308–9319.

POULOS, A. AND MAHONY, M. J. 2008. Effectiveness of feedback: The students' perspective. *Assessment & Evaluation in Higher Education 33,* 2, 143–154.

RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., SUTSKEVER, I., ET AL. 2019. Language models are unsupervised multitask learners. *OpenAI blog 1,* 8, 9.

ROSCHELLE, J., LESTER, J., AND FUSCO, J. 2020. Ai and the future of learning: Expert panel report. https://circls.org/reports/ai-report.

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

SELLAM, T., DAS, D., AND PARIKH, A. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7881–7892.

TANG, G., MÜLLER, M., GONZALES, A. R., AND SENNRICH, R. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, Brussels, Belgium, 4263–4272.

TYTLER, R. 2020. Stem education for the twenty-first century. In *Integrated Approaches to STEM Education: An International Perspective*, J. Anderson and Y. Li, Eds. Springer International Publishing, Cham, 21–43.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Vol. 30. Long Beach, CA.

VIJAYAKUMAR, A. K., COGSWELL, M., SELVARAJU, R. R., SUN, Q., LEE, S., CRANDALL, D., AND BATRA, D. 2017. Diverse beam search: Decoding diverse solutions from neural sequence models. In International Conference on Learning Representations. *arXiv preprint arXiv:1610.02424*.

WANG, W., FRASER, G., BARNES, T., MARTENS, C., AND PRICE, T. 2021. Automated classification of visual, interactive programs using execution traces. In *Proceedings of the 14th International Conference on Educational Data Mining*, I.-H. S. Hsiao, S. S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 677–681.

WEITEKAMP, D., HARPSTEAD, E., AND KOEDINGER, K. R. 2020. An interaction design for machine teaching to develop ai tutors. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, R. Bernhaupt, F. Mueller, D. Verweij, and J. Andres, Eds. Association for Computing Machinery, Hononlulu, HI, 1–11.

WINSTONE, N. E. AND BOUD, D. 2022. The need to disentangle assessment and feedback in higher education. *Studies in Higher Education 47,* 3, 656–667.

WOODS, B., ADAMSON, D., MIEL, S., AND MAYFIELD, E. 2017. Formative essay feedback using predictive scoring models. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, S. Matwin, S. Yu, and F. Farooq, Eds. Association for Computing Machinery, Halifax, NS, Canada, 2071–2080.

XIAO, Y., ZINGLE, G., JIA, Q., SHAH, H. R., ZHANG, Y., LI, T., KAROVALIYA, M., ZHAO, W., SONG, Y., JI, J., ET AL. 2020. Detecting problem statements in peer assessments. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, A. N. Rafferty, C. R. Jacob Whitehill, and V. Cavalli-Sforza, Eds. International Educational Data Mining Society, Montreal, Canada, 704–709.

YOUNG, S. 2006. Student views of effective online teaching in higher education. *The American Journal of Distance Education 20,* 2, 65–77.

YUAN, W., LIU, P., AND NEUBIG, G. 2022. Can we automate scientific reviewing? *Journal of Artificial Intelligence Research 75*, 171–212.

YUAN, W., NEUBIG, G., AND LIU, P. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Vol. 34. Morgan Kaufmann Publishers Inc., online.

ZAHEER, M., GURUGANESH, G., DUBEY, A., AINSLIE, J., ALBERTI, C., ONTANON, S., PHAM, P., RAVULA, A., WANG, Q., YANG, L., AND AHMED, A. 2020. Big bird: Transformers for longer sequences. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Curran Associates Inc., Vancouver, Canada.

ZHANG, T., KISHORE, V., WU, F., WEINBERGER, K. Q., AND ARTZI, Y. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*. Addis Ababa, Ethiopia.

ZHI, R., MARWAN, S., DONG, Y., LYTLE, N., PRICE, T. W., AND BARNES, T. 2019. Toward data-driven example feedback for novice programming. In *Proceedings of the 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society, Montreal, Canada, 218–227.

ZINGLE, G., RADHAKRISHNAN, B., XIAO, Y., GEHRINGER, E., XIAO, Z., PRAMUDIANTO, F., KHURANA, G., AND ARNAV, A. 2019. Detecting suggestions in peer assessments. In *Proceedings of the 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society, 474–479.