ABSTRACT

JIA, QINJIN. Perceive, Generate, Evaluate: Automated Feedback Generation for Student Project Reports and The Efficient Evaluation Thereof. (Under the direction of Dr. Edward F. Gehringer).

Feedback is a prominent element of the pedagogical cycle and exerts a substantial influence on fostering academic achievements and student success. Appropriate feedback can help learners consolidate or adjust their comprehension of knowledge and assist them in achieving their learning objectives. Instructors who endeavor to provide quality feedback on student assignments, particularly open-ended and complex assignments, must commit considerable time and effort, which may lead to delays in delivering feedback to students and affect their learning outcomes. Consequently, there has been a surge of interest in designing automated feedback-generation systems to produce textual feedback for student artifacts. These feedback systems can contribute to the advancement of teaching practices, and facilitate improvement of current educational frameworks and methodologies.

In this thesis, we focus on an ambitious feedback-generation task: producing feedback for student project reports. This assignment type exhibits a notable level of flexibility in terms of both the report content and the feedback. Delivering feedback on project reports is exceptionally time-consuming and demands perceptive domain expertise. Thus, exploring the feasibility of automatic feedback production for student project reports is exceedingly valuable. In addition, a concomitant challenge with feedback generation is efficient evaluation thereof. An intuitive strategy for evaluating generation is to involve humans as judges, but it is extraordinarily costly, time-consuming, and prone to inconsistency. Thus, it is essential to investigate efficient evaluation approaches, as they are the exclusive means to compare different generation outputs in a practical, systematic, and extensive way.

This dissertation consists of two parts. In the first part, we describe our endeavors to construct automatic feedback-generation systems for student project reports. In particular, we focus on large language model (LLM)-based data-driven methods. In Study 1, we present a system, dubbed *Insta-Reviewer*, for automatically generating feedback on project reports. We discuss its strengths and limitations while highlighting several prominent challenges for future research. In Study 2, we conduct an in-class evaluation of Insta-Reviewer to understand the practical implications and limitations of the feedback system. We survey student and instructor opinions on LLM-generated reviews in an actual class, and their perspectives

contribute the insights towards enhancing and optimizing feedback-generation systems to better fulfill the needs of students and instructors in practical educational settings.

In the second part, we explore approaches for efficiently evaluating system-generated feedback. Specifically, we are interested in using a limited number of human-judgment labels to construct feedback-evaluation models. These models are built to mimic evaluators in assessing the quality of feedback across different dimensions. Study 3 introduces a BERT-based multi-task learning model to concurrently assess multiple quality dimensions (viz., containing suggestions, mentioning problems, and adopting positive tone) of feedback. Study 4 presents a paradigm termed "incremental zero-shot learning," aimed at using a more restricted number of labels, even zero labels, to assess feedback quality. Study 5 investigates another crucial quality dimension, "faithfulness", which measures the alignment of generated feedback with the information presented in project reports.

Perceive, Generate, Evaluate: Automated Feedback Generation for Student Project Reports and The Efficient Evaluation Thereof

by
Qinjin Jia

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2024

APPROVED BY:

Dr. Bita Akram	Dr. Noboru Matsuda
Dr. Munindar P. Singh	Dr. Edward F. Gehringer Chair of Advisory Committee

DEDICATION

To my beloved mom and dad ♡, for giving me life and unconditional love.

BIOGRAPHY

Qinjin Jia was born in Xi'an, China. In 2017, he earned a Bachelor's degree with First-Class Honours in Electronic Engineering from the University of Liverpool, United Kingdom. He was also awarded the William Henry McMenemey Memorial Prize, recognized as the second-most prestigious undergraduate award in the Faculty, by the resolution of the Senate of the University. After completing his undergraduate studies, he went to the United States. In 2019, he received a Master's degree in Computer Engineering from Boston University. Then, he embarked on his journey towards a Ph.D. at North Carolina State University.

His research lies at the intersection of natural language processing and advanced learning technologies. Under the supervision of Professor Edward F. Gehringer, he worked on projects that utilizes large language models to enhance the student learning experience, aiming to contribute to the future AI-powered educational ecosystem. Throughout his doctoral studies, Qinjin has published papers in esteemed conferences and journals, such as the Annual Meeting of the Association for Computational Linguistics (ACL), the Building Educational Applications (BEA), and the Journal of Educational Data Mining (JEDM).

ACKNOWLEDGEMENTS

The past four and a half years at North Carolina State University have been an unforgettable and invaluable experience for me. It has been an inspirational journey with countless memorable moments that have shaped my character. Along this path, I have experienced joys, faced setbacks, embraced enthusiasm, navigated through lows, gained confidence, and conquered anxiety. I recognize that I could not have made this without the unwavering support and encouragement of many individuals, to whom I am profoundly grateful.

First and foremost, I would like to extend my heartfelt gratitude to my esteemed Ph.D. advisor, Dr. Edward F. Gehringer, for his invaluable guidance and unwavering support throughout my doctoral studies. I consider myself incredibly fortunate and deeply appreciative to have had the opportunity to work closely with Dr. Gehringer. His immense knowledge and bountiful experience have profoundly impacted my growth as a researcher, shaping my expertise and aspirations in the field. Thank you so much, Dr. Gehringer!

I would also like to extend my sincere appreciation to the other members of my Ph.D. advisory committee, Dr. Bita Akram, Dr. Noboru Matsuda, and Dr. Munindar P. Singh, for their contributions in refining this dissertation. Their profound expertise and insightful feedback have significantly enriched this work. I am sincerely appreciative of the considerable time and effort they devoted to guiding me through the completion of this dissertation. Many thanks to Dr. Akram, Dr. Matsuda, and Dr. Singh, for your unwavering support!

Finally, I would like to express my deepest gratitude to my family and friends, for accompanying me throughout all of this, giving me unconditional support, love, and trust. Their unwavering belief in me has always been a constant source of strength, propelling me forward during challenging times. I love you, mom, dad, grandmas and grandpas! Thank you for nurturing me, teaching me to be kind, and supporting me in everything I do. I also want to thank all my friends. I feel truly fortunate to know each and every one of you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Motivation

Feedback, a pivotal component of the instructional cycle, exerts a considerable influence on promoting academic achievement and student success (Chickering and Gamson 1987; Branford et al. 2000; Hattie and Timperley 2007; Alharbi 2017; Kusairi 2020). Appropriate feedback can help learners consolidate or adapt their understanding of knowledge and reduce discrepancies between current performance and learning intentions (Hattie and Timperley 2007; Shute 2008; Ahea et al. 2016). Moreover, immediate feedback is typically more efficacious than delayed feedback, presumably because instant feedback is more likely to motivate students to stay on task and attain their learning objectives (Kulik and Kulik 1988; Young 2006; Evans et al. 2014; Race 2019). Nevertheless, providing quality feedback on student assignments, especially for open-ended and complex ones (e.g., project reports), requires a significant time commitment from instructors and can be arduous to accomplish in a timely manner (Keuning et al. 2018). Thus, researchers have sought to develop a variety of automated feedback-generation systems to create feedback on student artifacts.

Previous research has investigated automated feedback generation for various forms of student work, such as novice programming (Zhi et al. 2019; Marwan et al. 2021), short-essay writing (Woods et al. 2017; Botarleanu et al. 2018), and open-ended short answers (Basu et al. 2013; Malik et al. 2019). 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 developed multifarious automated feedback systems to facilitate student learning, demonstrating promising outcomes across modalities and domains (Cavalcanti et al. 2021). However, the majority of studies focus on relatively straightforward student tasks for which feedback-generation rules can be explicitly formulated. These methods are inadequate for numerous more intricate types of student assignments since it is practically impossible to create a massive collection of feedback-generation rules for such complex artifacts.

One of the most ambitious feedback-generation tasks is creating feedback for student project reports, as this assignment type is remarkably flexible in terms of both the report content and the feedback. Course projects are an essential part of many university curricula, especially in STEM[1] courses (Borowczak et al. 2015; Gehringer 2020). They can help students strengthen their understanding of theoretical knowledge and cultivate a host of skills that are increasingly important in the professional world (Mannix and Neale 2005; Gehringer 2020). However, delivering feedback on project reports is extremely time-consuming and requires insightful domain expertise. Many instructors can merely provide summative feedback on projects at the end of the term, which does not give students an opportunity to implement the advice and improve their learning. Therefore, it would be worthwhile to examine the viability of automatic feedback production for student project reports.

In addition, a problem that accompanies feedback generation is the efficient evaluation thereof. Evaluating system-generated feedback is in itself a considerable challenge, primarily because feedback generation is a highly open-ended task. For instance, an automated feedback system can generate multiple plausible reviews for the same student report, but these reviews can differ vastly. The most natural strategy for evaluating the quality of system-generated feedback is to involve humans (or domain experts) as judges. Nevertheless, human evaluation can be extraordinarily costly, time-consuming, and prone to inconsistency (Celikyilmaz et al. 2020; van der Lee et al. 2021). Thus, it is indispensable to explore efficient feedback-evaluation approaches since they are the only way to compare different generation strategies in a practical, comprehensive, and extensive manner.

[1]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).

In this dissertation, the first theme is to investigate the feasibility of automatic feedback generation for student project reports. In particular, we focus on large language model (LLM)-based data-driven methods, which have shown superior performance on various natural language generation (NLG) tasks. More importantly, they can be effortlessly generalized to new assignment types and domains, unlike conventional rule-based feedback systems. The second theme is to explore approaches for efficiently evaluating system-generated feedback. Specifically, we are interested in using a relatively small number of human-judgment labels to implement feedback-evaluation models. These models are built to mimic evaluators in assessing the quality of feedback in different dimensions, such as faithfulness and whether the feedback contains suggestions or problem statements.

1.2 Research Outline

We begin in Chapter 2 with a literature review of automated feedback generation. We also provide relevant background knowledge in that chapter. Chapter 3 describes the datasets used in this thesis and how we respect the privacy rights of our participants during the data collection process. The rest of the dissertation consists of the following two parts:

Part I – Feedback Generation for Student Project Reports

Part I focuses on the task of automated feedback generation for project reports. We implement feedback generation systems and investigate their limitations. Specifically,

In Chapter 4, we make a first attempt to assess the feasibility of generating feedback for student project reports. We present a data-driven system, dubbed *Insta-Reviewer*, for producing instant textual feedback on student project reports. We conduct a comprehensive evaluation of the system-generated feedback, and analyze the strengths and weaknesses of Insta-Reviewer. We also perform ablation experiments to understand the contribution of each component in the system. Finally, we conclude by discussing its limitations and highlighting several prominent challenges for future research.

In Chapter 5, we conduct an in-class evaluation of our feedback-generation systems, to understand the practical impact and limitations of system-generated feedback in real-world educational settings. We solicit student opinions through questionnaires and delve into instructor perceptions via a qualitative semi-structured interview. Their responses support the effectiveness of generated feedback in real classes, and our analysis contributes insights for future research on automated feedback systems, steering them towards better fulfilling the educational needs of students and instructors.

Part II – Automatic Evaluation of System-generated Feedback

Part II explores how to efficiently evaluate the feedback produced by the systems. We are interested in implementing models to automate the assessment process. Specifically,

In Chapter 6, we investigate the viability of mimicking human evaluators to assess the quality of feedback by employing large language models (LLMs) and multi-task learning (MTL). We introduce a BERT-based MTL model to simultaneously assess multiple quality dimensions (viz., containing suggestions, mentioning problems, and adopting positive tone) of feedback. We then present results on two datasets: one is a collection of system-generated feedback and the other is a set of human-written feedback. Finally, we analyze the experimental results and discuss limitations and future research.

In Chapter 7, we further research the feasibility of using a very limited number of labels, even zero labels, to assess the quality of feedback. We present a new paradigm, called incremental zero-shot learning (IZSL), to reduce the need for labels. The core idea is to treat the problem as a natural language inference (NLI) task. We perform the experiments on two datasets and present results using multiple metrics. We also implement a BERT-based classification model and compare our approach with it after training on the same data. Finally, we discuss the results and open questions to be explored.

In Chapter 8, we delve into another significant quality dimension – faithfulness. Despite the generated feedback being highly fluent, coherent, and plausible sounding, automated feedback systems are prone to hallucinating content that deviates from the input document. Thus, we analyze the types of hallucinated content in the feedback and introduce NLI-based and ChatGPT-based models for mimicking human evaluators to assess the faithfulness of generated feedback. Lastly, we discuss the results and conclude the assessment of faithfulness remains an ongoing research challenge.

Finally, Chapter 9 concludes the dissertation, summarizes the limitations of our studies, and provides some insightful discussion of prospective directions for future research.

1.3 Research Questions

In summary, this dissertation seeks to answer the following high-level questions:

1. **RQ1**: *Can we automatically generate feedback for student project reports? What are the potential deficiencies of the system-generated feedback?* This research question is addressed in Chapter 4, where we thoroughly discuss the methodology and evaluation conclusions of the proposed feedback-generation system, Insta-Reviewer.

2. **RQ2**: *Do students appreciate system-generated feedback in actual classes? What may hinder the deployment of the automated feedback system?* This research question is discussed in Chapter 5, where we conduct an evaluation in actual classes to discover the deficiencies that impede the actual deployment of the feedback system.

3. **RQ3**: *Can we build machine-learning models to mimic human evaluators in order to automate the evaluation of feedback?* This research question is investigated in Chapter 6, Chapter 7, and Chapter 8, where we present multiple LLM-based models for assessing the quality of feedback across different evaluation dimensions.

1.4 Contributions

The principal contributions of this dissertation are summarized as follows:

- We are among the first to research automated feedback generation for student project reports. We introduce an automated feedback system, dubbed Insta-Reviewer, and a framework for evaluating system-generated feedback. The experimental results demonstrate the feasibility of automatic feedback generation for the reports.

- We design multiple LLM-based models for efficiently evaluating system-generated feedback. We present a BERT-based MTL model, an incremental zero-shot learning model, and faithfulness-judgment models. The models can effectively mimic human evaluators in assessing the quality of feedback across different dimensions.

- We conduct an in-class evaluation of the feedback system to solicit authentic opinions from students and instructors. The evaluations reveal limitations of current methods and suggest directions for future research. We also collect multiple datasets of student reports and human-authored reviews to facilitate future research endeavors.

CHAPTER

2

LITERATURE

This chapter intends to provide readers with relevant background and a literature review. In Section 2.1, we commence with a retrospective review of prior research on automated feedback generation, which falls into two categories: expert-driven and data-driven methods. Section 2.2 then delves into data-driven approaches and provides background on the implementation of data-driven feedback systems. Specifically, we introduce the seq2seq paradigm and pre-trained Transformer-based seq2seq generation models. Following this, Section 2.3 then surveys evaluation of text generation, including content-overlap metrics, embedding-based metrics, and human-centered evaluation. After that, Section 2.4 presents the background of the “hallucination" issue in generated feedback. Finally yet importantly, Section 2.5 provides a discussion of ethical concerns related to feedback generation.

2.1 Automated Feedback Generation

In the field of education, feedback is defined as information provided by an agent (e.g., teacher, peer) about a learner’s performance or understanding, and it is one of the most significant influences on student learning and achievement (Hattie and Timperley 2007). Previous research has been devoted to designing a variety of automated feedback systems that provide feedback on various forms of student work, such as short-answer questions, essays, and programming problems. Although these efforts were not intended to provide feedback on student project reports, we review these studies to gain some insight. We can broadly categorize previous research on automated feedback generation into two groups based on the feedback-generation models (i.e., feedback engines) used in the studies.

2.1.1 Expert-driven feedback generation

Expert-driven methods (also referred to as rule-based methods) utilize a set of pre-defined rules that encode expert knowledge and experience to provide feedback. Narciss et al. (2014) presented an intelligent tutoring system for helping students learn mathematics. The system uses a set of pre-defined rules provided by domain experts to deliver feedback for students' answers to numerical or multiple-choice questions. For instance, one rule might be that if a student's answer is 2/6 instead of 1/3, the system will provide feedback advising the student to review the explanation about reducing equivalent fractions to the lowest denominator. Nagata et al. (2014) introduced an expert-driven approach for diagnosing preposition errors and producing feedback for students' English writing.

Despite the fact that expert-driven methods typically yield accurate feedback and are not data-hungry, they have three significant limitations. First, they are not suitable for dealing with complex, open-ended problems (e.g., generating feedback for students' project reports) because creating and maintaining a vast collection of expert-designed rules for such open-ended 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, particularly open-ended ones (Marwan et al. 2021). Third, expert-driven methods usually produce feedback that is limited to pre-designed templates or fixed expressions without dynamic explanations (Lu and Cutumisu 2021; Deeva et al. 2021).

2.1.2 Data-driven feedback generation

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 means of machine-learning or deep-learning algorithms (Deeva et al. 2021). For example, Lu and Cutumisu (2021) implemented several deep-learning 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 (LLMs), such as BERT (Devlin et al. 2019), BART (Lewis et al. 2020a), and GPT (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 thesis, we also employ such a pre-trained language model to exploit its ability to capture long-span dependencies in lengthy text and to take advantage of pre-training.

2.2 Implementation of Data-driven Feedback Generation

In this section, we provide background on implementing data-driven feedback generation. We formulate the task of feedback generation for student reports as a natural language generation (NLG) task. Specifically, we consider it a text-to-text generation problem, where the source text is a project report, and the target text is a feedback message. The objective of the task is to effectively model the underlying relations from the source text to the target text, so that it can generate credible target text for unseen source text. In the following, Section 2.2.1 first introduces the sequence-to-sequence (seq2seq) paradigm, which is the mainstream text-to-text generation framework. Then, Section 2.2.2 delineates seq2seq pre-trained language models, which are the state-of-the-art seq2seq implementations.

2.2.1 Sequence-to-sequence Paradigm

Sequence-to-sequence paradigm, also referred to as encoder-decoder framework, was originally developed for machine translation and then rapidly became the de facto practice for various text-to-text generation problems (Sutskever et al. 2014). As illustrated in Figure 2.1, this paradigm consists of an encoder and a decoder. The encoder reads the source sequence

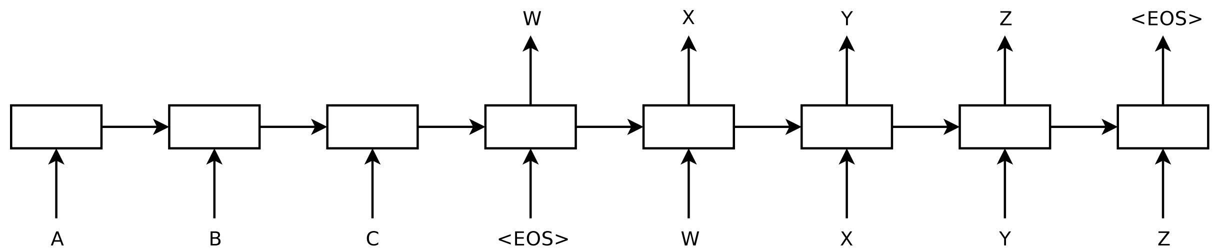

Figure 2.1: Sequence-to-sequence (seq2seq) paradigm. The model encodes an input sequence “ABC” into a numerical representation (also called the last hidden state), and then decodes the last hidden state into a output sequence “WXYZ.” <EOS> (end-of-sentence) represents the end of a sequence, which stops reading or producing (Sutskever et al. 2014).

and produces its representation, and then the decoder uses the source representation to generate the target sequence. These two components can be implemented using any kind of sequence-modeling architecture. Canonical choices are recurrent neural networks (RNNs) (Rumelhart 1986; Werbos 1990), long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber 1997), and gated recurrent unit networks (GRUs) (Cho et al. 2014).

However, a weakness of this framework is that information at the start of the sequence might be lost in the process of compressing source sequences into fixed representations, which is particularly challenging for long sequences. Thus, Bahdanau et al. (2014) proposed an attention mechanism that allows the decoder to have access to all source tokens at every decoding timestep. That is, the decoder can “pay attention” to relevant source tokens while producing the target sequence, instead of using only one representation of the source. Although the attention mechanism considerably mitigates the aforementioned issue, there is still a major drawback with using recurrent models (e.g., RNNs, LSTMs, and GRUs) for the encoder and decoder: the computations are sequential and cannot be parallelized.

2.2.2 Seq2seq Pre-trained Language Models

To parallelize the training and further eliminate the issue of long-span dependencies, Vaswani et al. (2017) proposed an encoder-decoder architecture named Transformer, which is entirely built upon the self-attention mechanism without using any recurrent layers. As shown in Figure 2.2, the Transformer follows the seq2seq paradigm and consists of two parts: the left part is an encoder, and the right part is a decoder. However, instead of reading and producing sequences recurrently, it relies entirely on the attention mechanism to draw

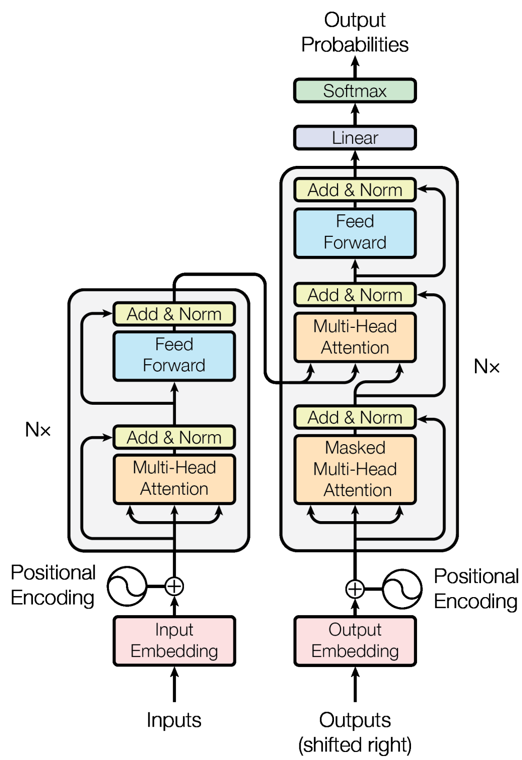

Figure 2.2: Architecture of the Transformer (Vaswani et al. 2017).

global dependencies. The self-attention mechanism encodes each word's relationship with every other word in the same sequence, paying more attention to the most relevant ones. The mechanism makes the model dynamically focus on certain parts of the sequence.

In parallel, Howard and Ruder (2018) proposed an effective transfer learning method named ULMFiT, and showed that training LSTM on a very large generic corpus first and then fine-tuning it on downstream tasks could considerably reduce the need for labeled data. By combining the Transformer architecture with transfer learning, researchers proposed a series of Transformer-based seq2seq models, such as T5 (Raffel et al. 2020) and BART (Lewis et al. 2020a), that eliminate the need to train task-specific architectures from scratch and can achieve superior performance on seq2seq tasks. However, such models that use the attention mechanism have quadratic time and space complexity, which limits the length of text that can be processed. The limited input length is a major challenge of our studies.

2.3 Evaluation of Feedback Generation

Appropriate metrics for evaluating generated feedback are essential since we need them to tune hyper-parameters, measure the progress, and reach meaningful conclusions. However, in contrast to other tasks (e.g., text classification), evaluation of generated text is largely an unsolved research problem, and there is no consensus on metrics for assessing text-generation systems (Van Der Lee et al. 2019). Accurately evaluating system-generated feedback (and many other NLG problems) is in itself a huge challenge (Celikyilmaz et al. 2020; Eddine et al. 2022), mainly because generating feedback is a highly open-ended task. For instance, a feedback system can generate multiple plausible reviews for the same student report, but all generated reviews can be vastly distinct in syntax and content.

All existing evaluation methods for NLG tasks can be grouped into three categories: 1) content-overlap metrics, 2) embedding-based metrics, and 3) human-centered evaluation metrics (Celikyilmaz et al. 2020). Content-overlap metrics automatically evaluate a text-generation system by measuring the word or n-gram overlap between generated texts and reference texts provided by domain experts. Embedding-based metrics evaluate a system by measuring the semantic similarity between generated texts and reference texts, which is achieved by mapping them to their embeddings and calculating the cosine distance between the embeddings. Human-centered evaluation asks people to assess the quality of system-generated texts against few task-specific criteria (e.g., tone and faithfulness).

It is worth noting that the ultimate goal of our automated feedback-generation systems 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; Eddine et al. 2022). Therefore, we also employ a content-overlap metric and an embedding-based metric in our studies to validate the outcomes of human evaluation. In the following paragraphs, we discuss potential evaluation metrics that can be applied to our task.

2.3.1 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 word overlap and 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 collection for tuning hyper-parameters or benchmark analysis with models since they are objective and fast to calculate (Celikyilmaz et al. 2020). 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, 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 scores in the harmonic mean. We report ROUGE-F_1 in results since both recall and precision are essential for our task.

2.3.2 Embedding-based metrics

Embedding-based metrics use learned representations of words and sentences to compute semantic similarity between generated and reference texts. Embedding-based metrics are generally more correlated with human judgment than the content-overlap metrics, but their behavior is not interpretable (Celikyilmaz et al. 2020). Given the exceptional performance of BERT (Devlin et al. 2019) across many tasks, recent studies on embedding-based metrics commonly leverage pre-trained contextual embeddings from BERT or similar pre-trained language models for evaluating the semantic equivalence between the texts (Sellam et al. 2020)). One of the most broadly used such metrics is BERTScore (Zhang et al. 2020).

BERTScore is an embedding-based metric proposed by Zhang et al. (2020) and has been shown to correlate well with human judgments for many text-generation tasks. BERTScore leverages the pre-trained contextual embeddings from BERT (Devlin et al. 2019) and matches words in generated and reference texts by cosine similarity. Then, it finds and picks the most similar words from the texts to calculate the score. 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 the semantic similarity between expert feedback and generated feedback.

2.3.3 Human-centered evaluation

Human-centered evaluation is the most natural way to evaluate the quality of feedback. However, it is expensive, time-consuming, and can be subjective. It asks human evaluators to judge the quality of generated text along some specific dimensions (e.g., readability and faithfulness). Caligiuri and Thomas (2013) found that positive tone and suggestions for improvement are key features of good feedback. Jia et al. (2021) mentioned that providing suggestions, mentioning problems, and using positive tone, are main characteristics of effective feedback. Celikyilmaz et al. (2020) pointed out that fluency and faithfulness are essential for evaluating system-generated text. In this thesis, we primarily evaluate system-generated feedback in four dimensions: Suggestions, Problems, Positive Tone, and Faithfulness. Specifics of these evaluation dimensions are described in Section 4.3.1.

2.4 Hallucinations in Feedback Generation

Despite the encouraging performance of the aforementioned LLM-based automated feedback systems, they inevitably encounter a number of challenges in practical applications, including hallucination (Ji et al. 2023), ambiguity (Tamkin et al. 2022), incompleteness (Zhang et al. 2023), under-informativeness (Gao et al. 2023), and bias (Navigli et al. 2023). Among these obstacles, hallucination is notably intractable and seriously impedes the actual deployment of the systems. In the following, Section 2.4.1 first presents the background of the hallucination issue (including its definition, categorization, and sources). Then, Section 2.4.2 surveys several potential methods for measuring hallucinations.

2.4.1 Definition, Categorization, and Sources

While the advancement of LLMs has rendered automatically generated feedback remarkably fluent, coherent, and akin to human-written feedback, LLM-based feedback systems are prone to generating hallucinated (i.e., unfaithful) content. In the following, we first provide the definition of the term hallucination. Then, we discuss different categorizations of hallucinations. Lastly, we summarize different factors that can induce hallucinations.

Definition of hallucination

Within the domain of text generation, *hallucination* is defined as an occurrence where the generated text is nonsensical or unfaithful (i.e., inaccurate) to the information presented in the source content (Filippova 2020; Maynez et al. 2020; Parikh et al. 2020b; Zhang et al. 2023). In simpler terms, the produced text can appear fluent, coherent, and plausible sounding, yet it could be inconsistent or not pertinent to the corresponding student submission.

In addition to hallucination, the literature encompasses two related concepts, namely *faithfulness* and *factuality*, which are sometimes used interchangeably or conflated (Ji et al. 2023). Both terms are antonyms of hallucination, referring to being actual or based on fact. However, they differ in what is considered as *fact*. For faithfulness, fact is the input source content, whereas for factuality, fact refers to real-world knowledge (Maynez et al. 2020). This dissertation emphasizes on the assessment of faithfulness rather than factuality.

Categorization of hallucinations

After defining hallucination and elucidating related concepts, we now further discuss two distinct types of hallucinations (Huang et al. 2021; Ji et al. 2023; Zhang et al. 2023).

Intrinsic Hallucinations are characterized by the generated text directly contradicting the input. This implies that the text is inconsistent with the input information. For instance, an output may erroneously claim "your work is missing a test plan," even though the student has already provided one in their submission.

Extrinsic Hallucinations, on the other hand, refer to cases where the faithfulness of generated text cannot be verified against source content (i.e., neither supported nor contradicted). For example, a model may fabricate text "your code screenshots have quite small text," despite the absence of any image in the input.

In addition, (Zhang et al. 2023) further classifies intrinsic hallucinations into input-conflict and context-conflict. The former refers to the generated content that contradicts the input, while the latter refers to the content that is internally self-contradictory. However, our work does not make this further distinction for the sake of simplicity and focus.

Sources of hallucination

Lastly, we explore potential factors that may induce hallucinations in text generation to better understand the issue.

Data is a primary cause of hallucination, which mainly refers to the lack of pertinent knowledge or internalization of false knowledge (Zhang et al. 2023). Inconsistencies, incompleteness, or biases in training data can directly lead to hallucinations. Moreover, LLMs are susceptible to misinterpreting spurious correlations, such as highly co-occurring associations, as faithful knowledge, leading to hallucinated outputs (Li et al. 2022b).

Training and Inference also contribute to hallucination (Parikh et al. 2020a), which refers to hallucinated generation stemming from the discrepancy in decoding between training and inference time, which is known as exposure bias (Ranzato et al. 2016). During training, the next token is predicted conditioned on the ground-truth prefix sequences, a method commonly referred to as teacher-forcing (Goyal et al. 2016)). However, during the inference, LLMs generate the next token conditioned on their previous generation.

2.4.2 Measuring Hallucinations in Generated Texts

Furthermore, measuring hallucinations in generated text is indispensable and serves two essential purposes. First, quantifying the level of hallucination empowers us to assess the quality of system-generated feedback in terms of its credibility and compare different feedback generation strategies. Second, identifying hallucinated content is the primary step towards mitigating hallucinations through the post-editing technique (Ji et al. 2023; Lei et al. 2023). Nevertheless, the free-form and open-ended nature of text generation poses a significant challenge in measuring hallucinations produced by LLM-based systems (Zhang et al. 2023). Therefore, we now proceed to survey potential measurement methods.

Reference–based methods

Reference-based methods refer to a collection of metrics that measure hallucinations by comparing generated texts with reference texts (e.g., ground-truth feedback provided by instructors) based on the surface-level content overlap, such as ROUGE (Lin 2004), or semantic similarity, such as BERTScore (Zhang et al. 2020). However, such methods necessitate reference texts and have been demonstrated to exhibit a limited correlation with human judgment (Dhingra et al. 2019; Maynez et al. 2020; Krishna et al. 2021).

Question answering–based methods

Question-answering (QA) based methods implicitly measure the consistency between generated texts and input texts (Zhang et al. 2023). These methods function under the assumption that if a piece of generated text is faithfully aligned with its source content, then similar answers should emerge from the same question (Durmus et al. 2020; Wang et al. 2020; Honovich et al. 2021). However, QA-based methods heavily contingent on the capability of the question-generation and question-answering models (Ji et al. 2023).

Natural language inference–based methods

Natural language inference (NLI) is the task of determining whether, given a premise, a hypothesis is true (entailment) or false (contradiction) (Yin et al. 2019). Such methods measure the faithfulness of generated outputs based on the hypothesis that the source content should logically entail the information in faithful generated text (Dušek and Kasner 2020; Maynez et al. 2020). Thus, if the NLI model tells us that the input content is likely to entail the output, we can conclude that the generated text is faithful and vice versa.

Off-the-shelf LLM–based methods

Off-the-shelf LLM-based methods refer to harnessing the capabilities of general-purpose LLMs (e.g., Meta's LLaMA (Touvron et al. 2023) and OpenAI's ChatGPT (Team 2022)) to serve as hallucination evaluators. Such methods involve using appropriate prompts to query LLMs to judge the faithfulness of generated text (Mündler et al. 2023; Lei et al. 2023). However, the effectiveness of such methods is highly dependent on the training corpora of LLMs and the quality of human-designed prompts (Zhang et al. 2023; Ji et al. 2023).

2.5 Ethical Concerns 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. (2022a) 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 offensive 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 (as presented in Chapter 4). 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 automated 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.

CHAPTER

3

DATA

In this chapter, we delve into the details of three datasets collected for this dissertation. Section 3.1 introduces the student project reports and feedback (SPRF) dataset, which is a collection of paired student project reports and instructor feedback. The SPRF dataset is gathered for testing the feedback-generation systems presented in Part I. Section 3.2 introduces the system-generated feedback (SGF) dataset, which contains a set of system-generated feedback for testing the feedback-evaluation models proposed in Part II. Finally, Section 3.3 shows the details of the peer-review feedback (PRF) dataset, which consists of thousands of student-provided reviews on peers' project reports. The syntactically diverse student reviews allow us to examine the potential of the feedback-evaluation models.

3.1 Student Project Reports and Feedback (SPRF) Dataset

This section introduces the SPRF dataset. First, we describe the data source in Section 3.1.1. Then, we explain how participants' privacy rights were respected during the data collection process in Section 3.1.2. Finally, we introduce statistics on the dataset in Section 3.1.3.

3.1.1 Data Source

The SPRF dataset is collected from CSC 517,[1] a graduate-level object-oriented development course at North Carolina State University. As an integral part of the course requirements, groups of two to four students work together on some course projects, either on assigned topics, or chosen from a list of potential projects provided by the course instructors. The dataset encompasses projects for which students engaged in activities such as refactoring code from an open-source Github project named Expertiza,[2] incorporating new features and functionalities, or developing automated unit tests for specific software modules.

Table 3.1: Sample human-authored reviews from the SPRF dataset.

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.

Among the course project deliverables, each team is required to submit a group report to document the work that has been completed, the methodologies that they have utilized, and other project-related material (e.g., how they tested their code). Such group reports are also referred to as wiki pages since they are added to the wiki maintained by the open-source project Expertiza. The instructor thoroughly reviews each of these wiki pages and provides textual feedback on each of them. To better understand our data, URLs to three anonymous and de-identified reports are provided in the footnotes.[3,4,5] In addition, Table 3.1 above displays three randomly sampled instructor reviews from the SPRF dataset.

[1] NCSU CSC517 Course Page: `http://go.ncsu.edu/csc517`
[2] Expertiza Github Page: `https://github.com/expertiza/expertiza`
[3] Sample 1: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%201.pdf`
[4] Sample 2: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%202.pdf`
[5] Sample 3: `https://anonymous.4open.science/r/EDM22-BF52/Student%20Report%203.pdf`

3.1.2 Privacy Protection

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

Table 3.2: Statistics on the SPRF 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 the BART model to create the generation function in our systems (detailed in Chapter 4).

# 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

3.1.3 Statistics on the SPRF 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 3.2 above summarizes statistics from the dataset, including the number of words and tokens of our data. Note that since the pre-trained large language model BART (detailed in Chapter 4) has a maximum input length limit of

1024 tokens,[6] we employ an unsupervised method to summarize original reports to lengths acceptable for input into BART. Statistics on the summarized reports are also offered.

In addition, we measured the average number of words and tokens for reports, summarized reports, and reference reviews (which are instructor or expert reviews) in the SPRF dataset. The average number of words for each original report is 1193, which corresponds to 1643 subword tokens. Notably, we found that 75.8% of the reports comprise more than 1024 tokens. Thus, we summarized (as detailed in Section 4.2.3) the original reports before feeding them into BART. The resulting average number of words in each summarized report is 704 (equivalent to 951 subword tokens). For the reference reviews, the average word count and the average count of subword tokens per review are 55 and 71, respectively.

3.2 System-generated Feedback (SGF) Dataset

This section presents the SGF dataset, which is a set of system-generated feedback. We collected the SGF dataset for testing the feedback-evaluation models in Part II. First, we describe the collection of the SGF dataset in Section 3.2.1. Then, we explain the annotation process in Section 3.2.2. Finally, we showcase statistics on the dataset in Section 3.2.3.

3.2.1 Data Collection

The SGF dataset is a collection of system-generated feedback produced by our feedback-generation system, Insta-Reviewer (as introduced in Chapter 4). The purpose of collecting this dataset is to examine our feedback-evaluation models, which are built to mimic human evaluators in judging the quality of feedback across different dimensions. Specifically, we generated a collection of feedback by training Insta-Reviewer multiple times with various hyper-parameters (viz., the batch size, the number of epochs, and the learning rate) in conjunction with different decoding algorithms and beam size. By this method, we obtained the SGF dataset consisting of 250 system-generated feedback messages. Table 3.3 presents three randomly sampled system-generated feedback messages from the SGF dataset.

[6] 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.

Table 3.3: Sample system-generated feedback from the SGF dataset.

Peer-Review Comments (lower-cased)	Sugg.	Prob.	Tone
This is a very good description of the changes that were made. The design doc is very readable. It describes the changes well. However, it does not explain how the changes were implemented. For example, the past-due assignments section is not shown.	✗	✓	✓
The writeup is quite good. It explains the changes well, but it could have been improved by adding more detail to the code. This would have helped the reader understand what was done.	✓	✗	✓
There are too many screenshots, and the code snippets should be separated out. Also, the tests are not described in any detail. They are just shown in a list. This is not very helpful. I would have liked to see the code sequences described in more detail.	✓	✓	✗

3.2.2 Annotation Process

First, we formally define each quality dimension (used to assess feedback) in the context of automated feedback evaluation as follows:

Suggestion: A feedback text is said to contain suggestions if it mentions how to correct a problem or make improvements.

Problem: A feedback text is said to detect problems if it points out something that is going wrong in the project report.

Positive Tone: A feedback text is said to use positive tone if it has an overall positive semantic orientation.

Based on the definitions, one annotator who is a fluent English speaker and familiar with the course context, annotated the dataset. For quality control, 50 reviews were randomly sampled from the dataset and labeled by a second annotator who is also a fluent English speaker and familiar with the course context. The inter-annotator agreement between the two annotators is measured by Cohen's κ coefficient, which is generally thought to be a more robust measure than the simple percent-agreement calculation (Cohen 1960). Cohen's κ coefficient for each dimension is shown in Table 3.4 below. According to Cohen's standard, the results indicate an almost perfect level of agreement between the two annotators.

Table 3.4: Inter-annotator agreement (Cohen's κ) in the SGF dataset.

Label	Suggestion	Problem	Positive tone	Average
Cohen's Kappa	0.94	0.92	0.91	0.92

3.2.3 Statistics on the SGF Dataset

Statistical information on the SGF dataset is showcased in Table 3.5 below. The dataset demonstrates a moderate level of imbalance. It consists of 250 system-generated feedback messages, with an average word count of 50 per feedback message. We observed that the majority of the system-generated feedback (over 85%) in the dataset contains problem statements and/or adopts positive tone. However, there are relatively fewer feedback messages (about 79%) that include suggestions. In addition, the statistical information reveals that feedback containing suggestions, problem statements, or adopting positive tone is, on average, approximately twenty words longer than feedback devoid of these elements.

Table 3.5: Statistics on the SGF dataset.

Label	Class	%samples	avg.#words	max#words
Suggestion	0	21.2%	33	59
	1	78.8%	56	105
Problem	0	14.8%	35	67
	1	85.2%	53	105
Pos. Tone	0	14.4%	36	86
	1	85.6%	53	105

3.3 Peer-review Feedback (PRF) Dataset

This section describes the PRF dataset, which is a set of peer-review feedback. Compared to the SGF dataset, the PRF dataset contains more syntactically diverse reviews, thus allowing us to conduct an in-depth study of the potential of the feedback-evaluation models proposed in Part II. First, Section 3.3.1 describes the data source. Then, Section 3.3.2 introduces the annotation process. Lastly, we showcase statistics on the dataset in Section 3.3.3.

Table 3.6: Sample rubric criteria used in the PRF dataset.

Does the design incorporate all of the functionality required?
Have the authors converted all the cases discussed in the test plan into automated tests?
Does the design appear to be sound, following appropriate principles and design patterns?

Table 3.7: Sample peer-review feedback from the PRF dataset.

Peer-Review Comments (lower-cased)	Sugg.	Prob.	Tone
Lots of good background details are given but the testing and implementation sections are missing.	✗	✓	✓
The explanation is clear to follow but it could also include some explanation of the use cases.	✓	✗	✓
Only problem statement is explained and nothing about design. Please add design and diagrams.	✓	✓	✗

3.3.1 Data Source: Expertiza

As introduced in Section 3.1, students enrolled in the graduate-level object-oriented development course, CSC 517, collaborate in teams to complete various course projects and are required to submit project reports. In addition to the feedback provided by the course instructors, their peers also contribute feedback based on the rubric criteria devised by the instructors. Specifically, we use a platform that supports multi-round peer review, allowing students to submit and revise their work, and engage in peer-reviewing other students' learning artifacts (e.g., project reports). In the assignments that provided the reviews for this dataset, two rounds of peer review (and one round of meta-review) were utilized:

1. The *formative-feedback* phase: For the first round of review, students upload substantially complete projects. The system then assigns each student to review a certain number of submissions and provide textual feedback, based on a rubric provided by the course instructors. Three sample rubric criteria are provided in Table 3.6.

2. The *summative-feedback* phase: After students have had an opportunity to revise their work based on feedback from their peers, final deliverables are submitted and peer-reviewed using a summative rubric. The rubric may include additional criteria such as "How well has the team addressed the feedback given in the first review round?". Many criteria ask reviewers to provide a numeric rating and textual feedback.

Table 3.8: Inter-annotator agreement (Cohen's κ) in the PRF dataset.

Label	Suggestion	Problem	Pos. Tone	Average
Cohen's Kappa	0.92	0.84	0.87	0.88

Table 3.9: Statistics on the PRF dataset.

Label	Class	%samples	avg.#words	max#words
Suggestion	0	79.2%	22	922
	1	20.8%	58	1076
Problem	0	56.7%	22	479
	1	43.3%	38	1076
Pos. Tone	0	22.2%	28	1040
	1	77.8%	29	1076

3. The *meta-review* phase: After the grading period is over, course instructors typically evaluate and grade (i.e. meta-review) the peer reviews provided by students.

All textual responses to the rubric criteria from both the formative-feedback phase and the summative-feedback phase were extracted to constitute the PRF dataset. Each response to a rubric criterion represents a peer-review message. After excluding reviews consisting solely of symbols and special characters, the dataset comprises a total of 12,053 samples. Table 3.7 above displays three randomly sampled peer-review feedback messages.

3.3.2 Annotation Process

The process of annotating the PRF dataset is similar to that of the SGF dataset, where one annotator who is a fluent English speaker and familiar with the course context annotated the dataset. Subsequently, a subset of 100 samples was randomly selected from the dataset and labeled by a second annotator, who also possesses fluency in English and is well acquainted with the course context. The agreement between the two annotators was measured by Cohen's κ coefficient (Cohen 1960). Table 3.8 above exhibits Cohen's κ coefficient for each dimension. Despite a slightly lower inter-annotator agreement compared with the SGF dataset, which might be attributed to the greater textual diversity in this dataset, the Cohen's κ coefficient still indicates a near-perfect level of agreement between the annotators.

3.3.3 Statistics on the PRF dataset

Statistical information on the PRF dataset is shown in Table 3.9. The minority class for each label includes more than 20% of the samples, and thus the dataset is mildly imbalanced. The average number of words in each peer-review feedback message is 29, which is about twenty words shorter than the SGF dataset. Our observations indicate that the majority (around 80%) of feedback messages adopt positive tone. Around half of the review comments contain problem statements, while only one-fifth of the reviews provide suggestions. In addition, compared with the system-generated feedback, the feedback in the PRF dataset exhibits a higher level of syntactic and semantic diversity, allowing us to test the potential of the feedback-evaluation models in a more comprehensive and insightful manner.

Part I

Feedback Generation for Student Project Reports

CHAPTER

4

A Data-driven Approach for Generating Feedback on Student Project Reports

4.1 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 encourage 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 unintentionally position students as passive recipients of feedback and limit their engagement with learning (Carless et al. 2011).

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. 2019; Basu et al. 2013). For instance, Malik et al. (2019) proposed generative models for providing feedback on short answers and programming assignments. Marwan et al. (2021) designed a hybrid method to deliver instant feedback for block-based programming. These and plenty of other impressive studies (e.g., (Lu and Cutumisu 2021; Orr and Russell 2021; Wang et al. 2021; Ariely et al. 2020; Mallavarapu 2020)) have designed multifarious feedback systems to facilitate student learning and have shown promising results across modalities and domains. It can be argued that automatic feedback systems will be integral components of the future AI-powered educational ecosystem (Roschelle et al. 2020).

However, to the best of our knowledge, no attempt has been made to evaluate the feasibility of automatic feedback generation on student project reports. It is well-known that course projects are an essential part of many university curricula, especially STEM 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. We summarize the reasons why such an automated feedback system for student project reports is significant as follows:

1. Despite the fact that course projects have many positive educational impacts on students, the burden of providing timely feedback may prevent instructors from offering sufficient course projects. In this case, an automated feedback system can encourage instructors to provide more project work in classes and benefit students.

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

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

RQ1: How effective is Insta-Reviewer in 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 student project reports can achieve promising results. 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-faithful or ambiguous statements in generated feedback and is inept at providing suggestions. In conclusion, our work demonstrates the feasibility of automated instant-feedback generation on student project reports. The experimental results also highlight several major challenges for future research.

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

The remainder of this chapter is organized as follows: Section 4.2 elaborates on our methodology for automatically generating feedback for student project reports. Section 4.3 describes our experimental setup and explains the evaluation metrics, including a new human-evaluation metric. Section 4.4 presents and discusses our experimental results. Section 4.5 introduces two ablation experiments to understand the contribution of each component in our system. Section 4.6 concludes the chapter, mentions the limitations of our work, and provides some discussion about future research.

4.2 Methodology

In this section, we detail our data-driven approach for automatically generating feedback on student project reports. We first formally define our task in Section 4.2.1. Then, we present the overall design of our feedback-generation system in Section 4.2.2. After that, Sections 4.2.3-4.2.4 elaborate on all components of the system.

4.2.1 Problem Formulation

We formulate the task of automatic feedback generation for student project reports as a *text-to-text generation* problem, where the source text is a long project report and the target text is a feedback message. 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 $\mathscr{V}$.

Then, the problem can be formally described as:

$$Y = \mathscr{F}_{\mathscr{M}}(X, \mathbb{C}) \tag{4.1}$$

where the model, or generation function, $\mathscr{F}_{\mathscr{M}}$ 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 $\mathscr{F}_{\mathscr{M}}$ in a data-driven manner using the dataset D, so that it can generate plausible and credible feedback for unseen student project reports. In this work, the generation function $\mathscr{F}_{\mathscr{M}}$ is crafted based

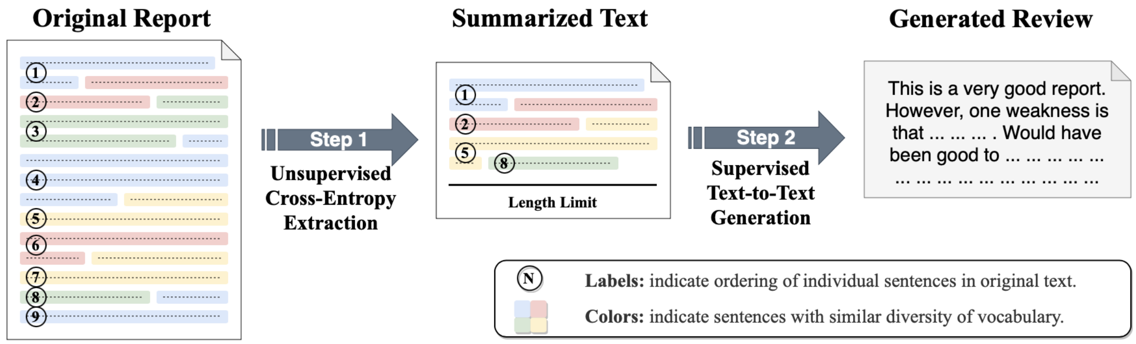

Figure 4.1: Operation of our *Insta-Reviewer* System. The system uses a select-then-generate paradigm (Pilault et al. 2020; Gehrmann et al. 2018; Yuan et al. 2022). The first step is to extract salient sentences (within the length limit) from students' project reports. The second step is to utilize a supervised NLP model to generate feedback for students' reports. Details of the system are described throughout Section 4.2.

on a large-scale pre-trained language model (LLM) called BART (detailed in Section 4.2.4), which has been demonstrated to be the state-of-the-art method to model the generation function for a variety of text-to-text generation tasks (Lewis et al. 2020a).

4.2.2 System Design

Despite the fact that the pre-trained language model BART is an effective method to model the generation function $\mathscr{F}_{\mathscr{M}}$, it has an input length limit of 1024 tokens[1] (Lewis et al. 2020a). Nevertheless, project reports are usually longer than that. In our dataset, 75.8% of the reports contain more than 1024 tokens, and the longest report is approximately eight times longer than that limit. One simple fix is to truncate the report by discarding all tokens beyond the length limit, but this can cause loss of critical information from the inputs (this hypothesis is verified in Section 4.5.1). Thus, we adopt a select-then-generate method (Gehrmann et al. 2018; Pilault et al. 2020; Yuan et al. 2022) to produce textual feedback on student project reports, which is illustrated in Figure 4.1.

Overall, the select-then-generate paradigm decomposes the problem into two sequential subproblems to resolve the issue of overlength input documents: 1) an unsupervised

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

sentence-level extractive summarization task, and 2) a supervised LLM-based text-to-text generation task. More formally, the problem description becomes,

$$Y = \mathscr{F}_{\mathscr{M}}(\mathscr{S}_{\mathscr{E}}(X), \mathbb{C}) \tag{4.2}$$

where the input to the generation function $\mathscr{F}_{\mathscr{M}}$ becomes $\mathscr{S}_{\mathscr{E}}(X)$, which represents a summarized report. The new function $\mathscr{S}_{\mathscr{E}}$ 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 $\mathscr{F}_{\mathscr{M}}$.

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 LLM BART (i.e., the implementation of the generation function $\mathscr{F}_{\mathscr{M}}$). In the second stage, we train the LLM BART on the summarized reports $\mathscr{S}_{\mathscr{E}}(X)$ and reference reviews Y. In the following sections (Section 4.2.3-4.2.4), we detail each stage of our feedback-generation system *Insta-Reviewer*.

4.2.3 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 (LLM) 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 summary S, as shown below,

$$Q(S) = -\sum_{w \in S} p_S(w) \log p_S(w) \tag{4.3}$$

$$p_S(w) = \frac{\textsc{Count}(w)}{\textsc{Len}(S)} \tag{4.4}$$

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.

$$Q(S) \; = \; \beta \cdot \textsc{Tokens}(S) - \sum_{w \in S} p_S(w) \log p_S(w) \tag{4.5}$$

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 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.

$$\gamma_t = Q(S)_{\lceil(1-\rho)N\rceil} \tag{4.6}$$

5. **Update** p: We use the sample values to update our probabilities, storing them as $\hat{p}_t$, according to the update rule below:

$$\hat{p}_{t,j} = \frac{\sum_{j=1}^{N} \delta_{[Q(S_j)\geq\gamma_t]}\delta_{[X_{i,j}=1]}}{\sum_{j=1}^{N} \delta_{[Q(S_j)\geq\gamma_t]}} \tag{4.7}$$

   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:

$$p_t = \alpha\hat{p}_t + (1-\alpha)p_{t-1} \tag{4.8}$$

7. **Termination**: If the value of γ_t has not changed in 3 iterations then the process terminates, returning the current p_t. Otherwise, it returns to step 3 and repeats.

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

4.2.4 Step 2: LLM-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 $\mathscr{F}_{\mathscr{M}}$ in Equation 4.2. We first introduce the BART model used for crafting $\mathscr{F}_{\mathscr{M}}$, then we describe a decoding algorithm called diverse beam-search (DBS) for improving the qulaity of generated feedback.

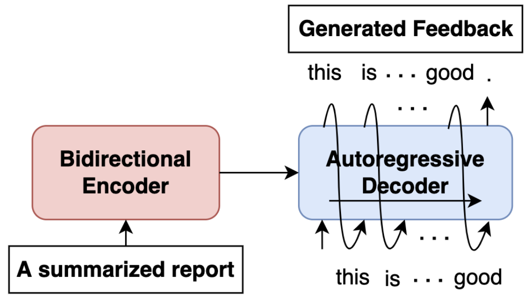

Figure 4.2: Illustration of an encoder-decoder architecture. The encoder can convert an input sequence of text (e.g., a summarized report) into a rich numerical representation, and then the decoder generates the output sequence (e.g., feedback) by iteratively predicting the most probable next word.

Modeling the Generation Function with BART

In this work, we employ a state-of-the-art LLM BART (Lewis et al. 2020a), which stands for bidirectional and auto-regressive transformers, to model the generation function $\mathscr{F}_{\mathscr{M}}$. The BART model is suitable for text-to-text generation tasks since it utilizes an encoder-decoder architecture (as illustrated in Figure 4.2), which can effectively model complex mappings from one sequence of text (e.g., summarized reports) to another (e.g., feedback).

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[2]" to initialize all parameters of the model and then fine-tune the model on a training set drawn from our dataset D.

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

[2]facebook/bart-large-cnn checkpoint: `https://huggingface.co/facebook/bart-large-cnn`

generate better feedback. In the original BART model setting, feedback is generated by iteratively selecting the word with the highest probability at each position in the sequence, which is referred to as greedy decoding. Greedily choosing the word with the highest probability at each step might be optimal at the current spot in the sequence, but as we move through the rest of the full sentence, it might turn out to be a non-optimal choice (output can be ungrammatical, unnatural, and nonsensical) since the greedy decoding algorithm lacks backtracking.

On the other hand, the DBS algorithm keeps track of the top-n most probable next words, where n is the number of beams. The next set of beams is chosen by considering all possible next-word extensions of the existing set and selecting the n most likely extensions. Additionally, in order to improve the diversity in the outputs, all beams are divided into groups, and diversity between the groups is enforced by the DBS algorithm. In our experiments, we set beam size to 4 and the number of groups to 2 since this combination yields the best results. DBS is applied to all models in this work, including the naïve BART model (Section 4.5.1) and the BigBirdPegasus model (Section 4.5.2).

4.3 Evaluation and Experimental Setting

In this section, we introduce our evaluation metrics and describe experimental settings. Section 4.3.1 presents a set of metrics, which includes a human evaluation metric, which includes a human evaluation metric, for evaluating generated textual feedback. Then, Section 4.3.2 introduces training details and our hardware setup.

4.3.1 Evaluation Metrics

We evaluate generated feedback with a comprehensive set of metrics, including a content-overlap metric ROUGE, a embedding-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.

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

Embedding-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[4] (Zhang et al. 2020).

Human Evaluation

After reviewing relevant manuscripts (e.g., Caligiuri and Thomas (2013); Zingle et al. (2019); Xiao et al. (2020); Jia et al. (2021)) and discussions with other researchers, we selected the following five dimensions to evaluate the feedback manually: Readability, Suggestions, Problems, Positive Tone, and Faithfulness. Our scores for these five manual-evaluation dimensions are calculated as follows:

1. ***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.

2. ***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

[3]Google rouge score package: `https://github.com/google-research/google-research/tree/master/rouge`

[4]BERTScore package: `https://github.com/Tiiiger/bert_score`

suggestion statement that can guide the reviewee in how to correct a problem or make improvements. Otherwise, we give a score of 0.

3. ***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.

4. ***Positive Tone*** (TONE): Feedback phrased in 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.

5. ***Faithfulness*** (FACT): The statements in generated feedback should stay consistent and truthful to the provided report. Otherwise, they may inadvertently mislead the reviewee and negatively impact learning. Faithfulness is calculated as:

$$\text{FACT} = \frac{\text{Count(consistent and truthful statements)}}{\text{Count(total statements)}}$$

   where the numerator is the total number of faithful (i.e., consistent and truthful) statements, and the denominator is the total number of statements in the feedback. If the denominator is 0, we directly give a score of 0.

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

4.3.2 Experimental Setup

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.

Hardware Setup

The BART models are trained on an NVIDIA RTX6000 GPU (24GB). The BigBirdPegasus model (mentioned in Section 4.5.2) 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.

4.4 Experimental Results and Discussion

4.4.1 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 4.1.

Experimental Results from Study 1

In the first row of Table 4.1, 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 4.3.1, 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 4.1, 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 4.2, and their corresponding evaluation scores are shown in Table 4.3.

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 4.2 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 of the "generated feedback 1" and the "expert feedback 1", we can find

Table 4.1: 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. READ=Readability, SUGG=Suggestions, PROB=Problems, TONE=Positive Tone, and FACT=Faithfulness.

	Method	ROUGE			BRTS	Human Evaluation					
		R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT	Average
	1. Expert Feedback	-	-	-	-	**100.0**	**82.0**	90.0	93.0	**100.0**	**93.00**
Insta-Reviewer	2. CE + BART	**28.54**	**6.39**	**18.21**	**59.18**	94.0	66.0	**96.0**	**95.0**	72.9	84.78

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 system-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%, 27.1% for “Readability,” “Suggestions,” and “Faithfulness,” respectively. The results imply that although our system is not as good as experts at providing suggestions and may produce some non-faithful or ambiguous statements, it is good at generating fluent and positive feedback that mentions problems that need to be addressed.

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-faithful or ambiguous statements.

More specifically, we observed from Table 4.1 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. Future 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 this thesis 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-faithful statements). Future work could therefore attempt to alleviate the drawbacks by combining data-driven and expert-driven approaches.

4.4.2 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 faithful, we found four potential imperfections in the system-generated feedback upon inspection. Table 4.2 shows three randomly sampled system-generated reviews, and Table 4.3 displays their corresponding ROUGE scores, BERTScore, and human evaluation scores.

Experimental results from Study 2

In Table 4.2, 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 faithful.

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 4.2. The generated feedback also detected that the report did not describe the test plan. However, the "generated feedback 2" contains a non-faithful statement (italicized): "…there is no description

Table 4.2: Sample expert and generated feedback. Evaluation scores are shown in Table 4.3. The generated feedback may contain non-faithful 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 4.3: ROUGE F_1 (with porter stemming), BERTScore, and Human Evaluation scores (%) for the sample generated feedback shown in the Table 4.2. † Note that the statement "I would suggest that the reader read the code ..." is not a valid suggestion for the reviewee.

	ROUGE			BERTScore	Human Evaluation					
	R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT	Average
Test set average (n=50)	28.54	6.39	18.21	59.18	94.0	66.0	96.0	95.0	72.9	84.78
Generated Feedback 1	27.52	1.87	20.18	59.17	100.0	100.0	100.0	100.0	100.0	100.0
Generated Feedback 2	45.09	18.00	21.57	66.79	100.0	0.00†	100.0	100.0	83.33	76.67
Generated Feedback 3	22.22	0.00	13.89	52.92	80.00	0.00	100.0	100.0	80.00	72.00

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 "Faithfulness," 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-faithful statement - "the changes are described well." and a vague indicator pronoun ("they"). From the third row of Table 4.3, 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.

Discussion of Study 2

In the second experiment, we manually inspected all system-generated feedback and found four potential deficiencies: non-faithful 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 generated feedback.

First, the automated feedback system may occasionally (≈27.1% of all statements) generate some non-faithful or ambiguous statements in the feedback. These non-faithful 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-faithful 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 4.2 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" statements. 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 faithfulness of the feedback generation. Increasing the beam size of the decoding algorithm (i.e., the diverse beam search algorithm introduced in Section 4.2.4) would improve the faithfulness 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.

4.4.3 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 4.4.

Experimental results from Study 3

In the first, the second, and the third row of Table 4.4, we show ROUGE scores and BERTScor (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 4.4, 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 4.4: ROUGE F_1 (with porter stemming) and BERTScore on the test set (n=50) in small-data settings (≤ 100 training samples). For each score, we report a 95% confidence interval (CI).

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

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.

4.4.4 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 2.5, 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.

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.

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.

4.5 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 (LLM) 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 4.5. 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 4.5.1 and 4.5.2, respectively.

Table 4.5: ROUGE F_1 (with porter stemming), BERTScore, and Human Evaluation scores (%) on the test set (n=50). All our ROUGE scores have a 95% confidence interval (CI) of at most ±2.98 as reported by the official ROUGE script (bootstrap resampling). Our BERTScore has a 95% CI of at most ±1.54. The "facebook/bart-large-cnn" checkpoint was used to calculate BERTScore.

	Method	ROUGE			BS	Human Evaluation					
		R1	R2	RLsum		READ	SUGG	PROB	TONE	FACT	Average
	1. Expert Feedback	-	-	-	-	**100.0**	**82.0**	90.0	93.0	**100.0**	**93.00**
Insta-Reviewer	2. CE + BART	**28.54**	**6.39**	**18.21**	**59.18**	94.0	66.0	**96.0**	**95.0**	72.9	84.78
Ablation Exp 1	3. naïve BART	25.88	4.98	17.23	58.03	82.0	58.0	94.0	93.0	66.7	78.74
Ablation Exp 2	4. BigBirdPegasus	15.92	4.04	13.45	-	56.0	2.0	12.0	58.0	23.4	30.28

4.5.1 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 4.2.2, 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 student 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 4.5, 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 4.5, "CE + BART" significantly ($\geq$ 6%) outperforms the "naïve BART" method on the "Readability," "Suggestions," and "Faithfulness" 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 ($\geq$ 6%) on the "Readability," "Suggestions," and "Faithfulness" 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.

4.5.2 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 LLM 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[5]” 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 4.5, 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 49.5% for “Readability,” “Suggestions,” “Problems,” “Positive Tone,” and “Faithfulness,” 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.

[5]google/bigbird-pegasus-large-arxiv checkpoint:
`https://huggingface.co/google/bigbird-pegasus-large-arxiv`

4.6 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 chapter, 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 promising results and even outperform human experts in the "Problems" and "Positive Tone" dimensions. However, the system may occasionally generate some non-faithful 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.

4.6.1 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 efficient 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.

4.6.2 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-faithful 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-faithful statements. Thus, an important direction for future work is to investigate how to avoid generating non-faithful statements in feedback.

The problem of faithfulness has two potential solutions. The first promising way is to automatically evaluate the correctness of each sentence in generated feedback and remove non-faithful 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-faithful. Future work can explore further along this direction. The other possible method to address the problem of non-faithful 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.

CHAPTER

5

Automated Feedback in Real Classes and Beyond: Perspectives from Students and Instructors

5.1 Introduction

In the realm of education, feedback refers to information provided by individuals (e.g., instructors or peers) regarding students' performance in learning activities (Alharbi 2017; Hattie and Timperley 2007; Race 2001). Feedback is integral in guiding students through their learning process, offering insights that enable them to strengthen or correct their understanding of knowledge and content (Hattie and Timperley 2007; Kusairi 2020; Race 2001). However, providing quality feedback, especially for assignments that lack straightforward answers, requires considerable effort and educational resources and often faces challenges in being delivered promptly. The demand for immediate and cost-efficient feedback solutions has driven the development of automated feedback systems (Jia et al. 2022a;

Kasneci et al. 2023). For example, Chapter 4 introduced Insta-Reviewer, an LLM-based automated feedback system for producing timely feedback on student project reports.

However, most LLM-based automated feedback systems have not been deployed in actual classroom settings, thus their applicability in real-world contexts remain unclear (Dai et al. 2023; Jia et al. 2022a,b; McNichols et al. 2023; Yoon et al. 2023; Han et al. 2023). These studies often employ automatic metrics (e.g., ROUGE scores (Lin 2004), BERTScore (Zhang et al. 2020)) and human judgment to evaluate the feedback across different quality dimensions (e.g., readability). Although these evaluations may reflect some aspects of the quality of generated feedback, they lack perspectives from students and teachers in real classrooms (Garbacea and Mei 2020). Consequently, their evaluation may not fully align with or comprehensively represent the authentic user experience in real educational contexts. This gap raises the question of whether these automated feedback systems can function effectively in actual classes and provide assistance to students and instructors.

In this Chapter, our primary objective is to gain a comprehensive understanding of the potential impact and limitations of automated feedback systems from perspectives of students and instructors in practical educational settings. To this end, we deploy the automated feedback system Insta-Reviewer to generate textual feedback on student project reports in a graduate-level computer science course. We then solicit student opinions on the generated feedback through a questionnaire, and engage with the course instructor to delve into their perspectives regarding the alignment of the feedback with their pedagogical objectives and methodologies. Through these, we aspire to contribute the insights derived from this study towards enhancing and optimizing feedback-generation systems to better fulfill the needs of both students and instructors in real-world educational scenarios.

To understand the impact and limitations of LLM-generated feedback in real educational settings, our work seeks to answer the following research questions (RQs):

RQ1: How do students and instructors perceive the system-generated feedback in actual classrooms? Do they find the system-generated feedback helpful for student learning?

RQ2: What limitations do students and instructors identify with the system-generated feedback? What are their specific concerns regarding the system-generated feedback?

RQ3: What insights can be drawn from the perspectives of students and instructors? What aspects of LLM-based automated feedback systems can be improved in the future?

The main conclusions are as follows: Both students and instructors find the system-generated feedback to be valuable, yet highlight some areas requiring enhancement. First, a major concern is the presence of unfaithful content in the feedback (i.e., prose that is erroneous, misleading, or entirely irrelevant to the student work). Second, students indicate a preference for more specific feedback tailored to their individual projects, rather than generic comments. Third, there is a consensus that automated feedback, while helpful, cannot fully replace the nuanced insights offered by instructors. Lastly, students express a desire for an interactive feedback system that can address their follow-up questions.

Our primary contributions are: 1) we conduct an empirical study that deploys an LLM-based automated feedback system in a graduate-level computer science course, collecting authentic user experiences that support the effectiveness of generated feedback in real classroom settings; 2) we collect perspectives from students and instructors that highlight the strengths of the generated feedback in actual classes while also revealing limitations such as the presence of unfaithful content and lack of project specificity; 3) our analysis contributes insights for future research on automated feedback systems, steering them towards better fulfilling the needs of students and instructors in actual classrooms.

The rest of the chapter is organized as follows: Section 5.2 first outlines the methodology and procedures employed in this study, including research setup, how feedback is generated, questionnaire design, and interview protocol. Subsequently, Section 5.3 presents the survey results and interview summary. After that, Section 5.4 analyzes perspectives from students and instructors, and discusses four insights drawn from their opinions. Finally, Section 5.5 concludes this chapter, highlights the limitations, and discusses about future studies.

5.2 Study Design

5.2.1 Study Setup

This study was approved by the IRB (institutional review board) and conducted in CSC 517, an object-oriented graduate-level course at North Carolina State University. A total of 82 students participated in the research. They were divided into 28 groups to work on course projects. The projects covered in this study involved students engaging in activities such as refactoring code, integrating new features and functionalities, or implementing automated unit tests for software modules. For the course-project deliverables, each group was required to submit documentation of the work completed, the methodologies they utilized, and other project-related materials, such as the procedures for testing their programs.

Table 5.1: Samples of system-generated feedback for student project reports.

Generated Feedback 1: "The writeup is quite readable, and it explains the rationale for the design of the project in a readable way. The design doc is divided into sections, with sections on the design, implementation, and testing. It would have been better to describe the functionality in more detail, e.g., how it works, and what changes are made to the code. In general, the design document is quite good."
Generated Feedback 2: "Good job of describing the changes to be made, and explaining why they were made. Would have been helpful to have more detail on the code changes, e.g., what was changed, and why. It would be better to link to the code in Github, where it would be easier to see what was changed. Also, it would have been good to have a summary of what was done, rather than just a list of changes."
Generated Feedback 3: "The writeup is quite readable, and the changes are described in detail. The code changes are clearly described, and they are described with sufficient detail. It would have been better to link to the Github diff view to show the code that has been changed. Also, I would have liked to see more explanation of the tests, e.g., what are the tests for late_policy_spec.rb, and why are they needed?"

The overall process was as follows: students first uploaded near-finalized projects, upon which their peers provided feedback based on rubric criteria devised by the instructor. Following the peer feedback, students revised their reports and submitted the final versions. Subsequently, an automated feedback-generation system produced feedback for the final reports and disseminated it to the students and the instructor via email. The questionnaire was simultaneously sent to each student, who was asked to complete it within one week. At the same time, the instructor examined the project reports and provided feedback. After the instructor checked all reports, we conducted an interview with the instructor.

Specifically, we generate feedback for student project reports by utilizing Insta-Reviewer, a BART-based data-driven automated feedback system, introduced in Chapter 4. The BART model is an encoder-decoder LLM (Lewis et al. 2020b), which is adept at capturing relationships from one sequence of text (e.g., project reports) to another (e.g., textual feedback). We fine-tuned the model with 484 pairs of past project reports and instructor feedback. In addition, recognizing the absence of effective automatic evaluation metrics for text generation (Celikyilmaz et al. 2020), we generated three sets of feedback by leveraging different combinations of decoding strategies and hyperparameters, and manually selected the best feedback from them. Table 5.1 above exhibits three samples of the generated feedback.

Table 5.2: All questions in the questionnaire, their average scores (Avg.) and standard deviations (SD).

Question (Scale)	Question Description	Avg.	SD
Q1. Overall Score (1–10)	Please provide an overall score for the feedback.	8.14	2.28
Q2. Helpfulness (1–5)	How helpful do you think the feedback is?	4.04	1.21
Q3. Faithfulness (1–5)	How accurate do you think the feedback is?	3.89	1.14
Q4. Comprehensiveness (1–5)	How comprehensive do you think the feedback is?	4.25	1.02
Q5. Replaceability (1–5)	Do you think the generated feedback can replace the instructor feedback?	3.25	1.45
Q6. Open Question	Please give your detailed comments on the feedback.	–	–

5.2.2 Survey Design

To gather student opinions on the system-generated feedback for their project reports, we designed a questionnaire with a series of Likert scale questions, followed by an open-ended free-text response. In constructing the questionnaire and formulating the questions, we adhered to the guidance outlined in Kitchenham and Pfleeger (2008). Additionally, we opted for an anonymous survey to increase response rates, encourage honest responses, and mitigate privacy concerns (Scherer et al. 2013; Tyagi 1989). Students in the class were informed, both verbally and in writing, that participation in the survey was entirely voluntary and would not affect their grades. Their responses were collected using Google Forms, and the results were securely stored on a cloud service administered by our institution.

After the IRB information and consent form at the beginning of the questionnaire, each survey consists of six questions. The specific survey questions are listed in Table 5.2. The first question (Q1) asks students to provide an overall rating of the feedback. Subsequently, Q2–Q4 engage students in assessing the feedback from the perspectives of helpfulness, faithfulness, and comprehensiveness, respectively. Q5 probes whether students perceive the system-generated feedback as an effective substitute for instructor feedback. Finally, Q6 is an open-ended free-text response question that allows students to offer explanations for their previous ratings or suggestions for future improvements of the feedback.

5.2.3 Interview Protocol

The insights of the course instructor are of paramount significance in comprehending the practical implications of the LLM-generated feedback within an educational framework. We followed the guidance outlined in (Kallio et al. 2016), and conducted a qualitative semi-structured interview with the course instructor to delve into his nuanced perspectives. The semi-structured interview offers a balanced approach that combines structure with flexibility, allowing for thorough exploration of topics while accommodating the unique perspectives and experiences of the instructor. This interview was conducted at the end of the course, after the instructor had reviewed all project reports and generated feedback.

This interview is divided into three sections. We first elicited the instructor's holistic impressions of the LLM-generated feedback. Second, we proceeded to delve into several specific aspects of the feedback, including its helpfulness, faithfulness, specificity, and comprehensiveness. Finally, we discussed with the instructor regarding his perspectives on how to integrate automated feedback systems into existing pedagogical frameworks and methodologies, as well as explored potential synergies or challenges that may arise in real-world applications. Throughout the interview process, we also engaged in impromptu discussions to probe further into viewpoints and insights mentioned by the instructor.

5.3 Empirical Results

5.3.1 Survey Results

For our survey, we received a total of 28 responses, with 23 of them providing detailed textual comments. The average score and standard deviation for each Likert scale question are presented in Table 5.2. The distribution of scores for the Likert scale questions can be found in Figure 5.1–5.5. Representative snippets of textual comments are shown interspersed throughout the following analysis to illustrate the results.

Overall, responses to Q1, as shown in Figure 5.1, indicate a prevailing satisfaction with the LLM-generated feedback. The majority of students (75.0%) rated the feedback 8 or above out of 10. Students commended the feedback for its fluency, coherence, and effectiveness in guiding them to enhance their project reports.

- *"The feedback is detailed and very helpful as an initial guidance to improve our report for the final project."*

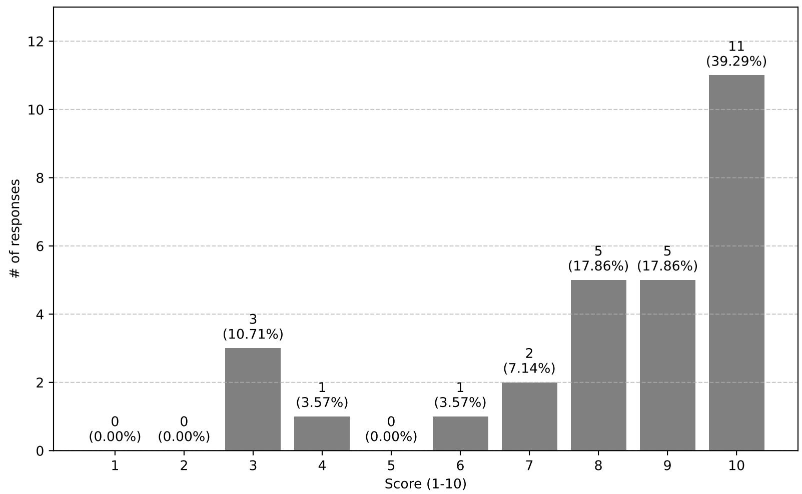

Figure 5.1: The distribution of scores for "Q1 – Please provide an overall score for the feedback." Scale 1 (bad) – 10 (good), mean=8.14, SD=2.28, n=28 groups and 82 students.

- *"It is very impressive. It is almost like a real person's feedback. I found the feedback accurate to some extent. However, some part of the feedback was not clear."*

The responses to Q2 (Helpfulness), as displayed in Figure 5.2, suggest that students perceive the LLM-generated feedback as helpful, with most students (78.6%) rating it as 4 or higher out of 5. However, student comments suggest that the primary factor diminishing the helpfulness of the feedback is lack of project specificity.

- *"It suggested me some changes for the code comparison addition which seemed helpful but I felt like the feedback seemed to be generic rather than document specific."*
- *"Some of the feedback I have received is not very detailed, while others are pretty descriptive."*

The answers to Q3 (Faithfulness), as shown in Figure 5.3, indicate that students have a concern about the faithfulness of the feedback. The average score for this question is 3.89 out of 5, and 67.9% of students scored it as 4 or above. Students perceive that such unfaithful content undermine the reliability of the feedback.

- *"The feedback said 'Include screenshots' and 'More explanation of the tests.' But they were already in place."*

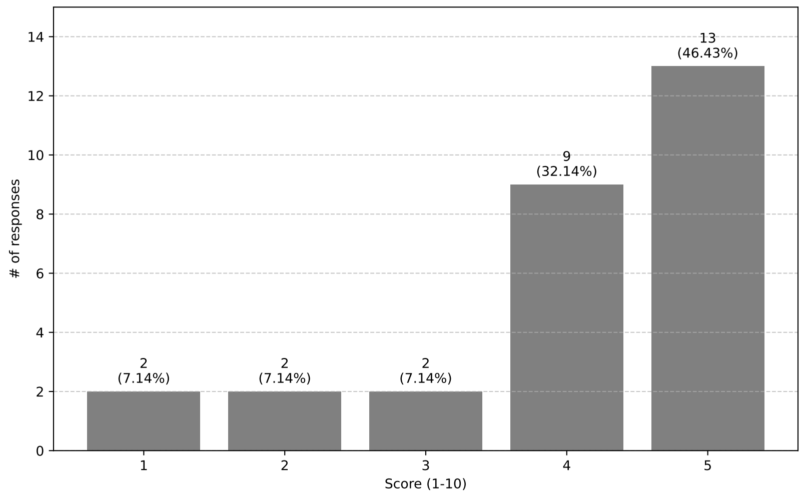

Figure 5.2: The distribution of scores for "Q2 – How helpful do you think the feedback is?" Scale 1 (not helpful) – 5 (very helpful), mean=4.04, SD=1.21, n=28 groups and 82 students.

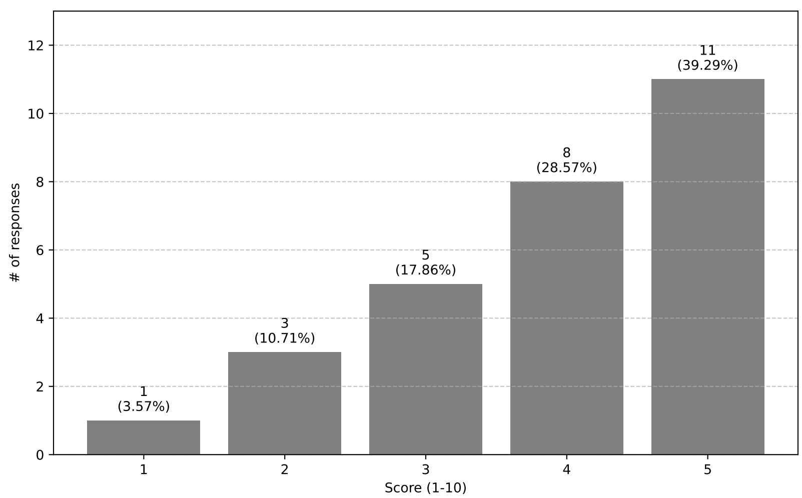

Figure 5.3: The distribution of scores for "Q3 – How accurate do you think the feedback is?" Scale 1 (not accurate) – 5 (very accurate), mean=3.89, SD=1.14, n=28 groups, 82 students.

The responses to Q4 (Comprehensiveness), as illustrated in Figure 5.4, demonstrate satisfaction with the comprehensiveness of the feedback. Nearly all students (92.9%) rated it as 4 or higher out of 5. However, the results may be biased as students may not be aware

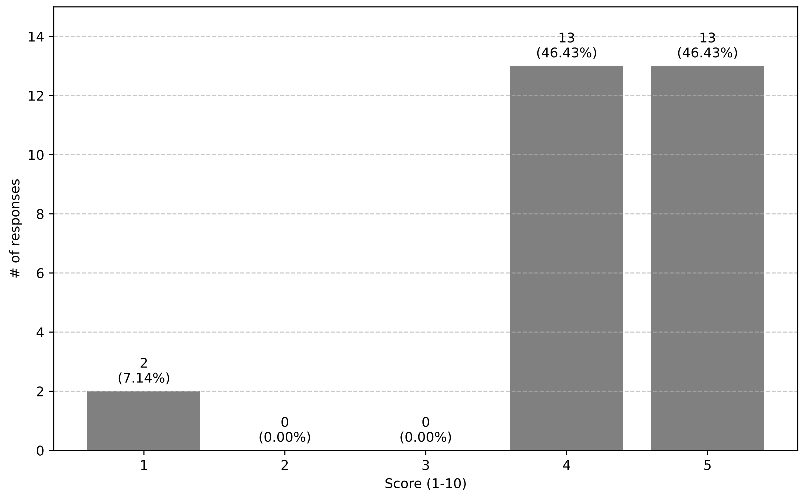

Figure 5.4: The distribution of scores for “Q4 – How comprehensive do you think the feedback is?” Scale 1 (not comprehensive at all) – 5 (very comprehensive), mean=4.25, SD=1.02, n=28 groups and 82 students.

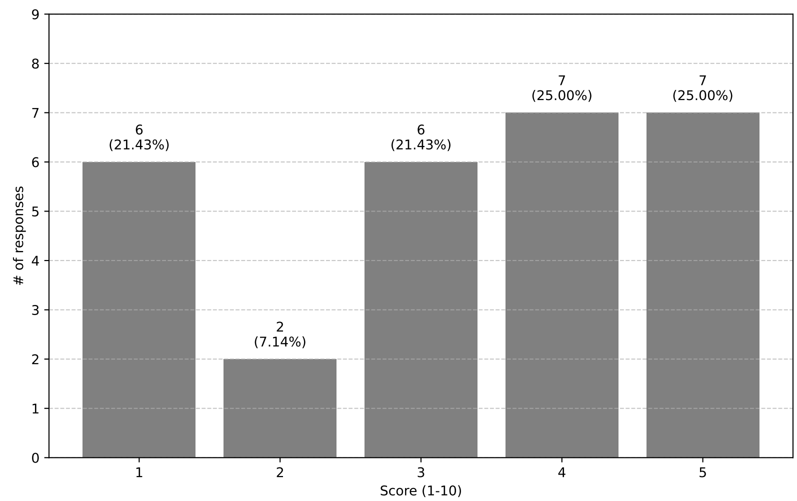

Figure 5.5: The distribution of scores for “Q5 – Do you think the system-generated feedback can replace the instructor feedback?” Scale 1 (not likely) – 5 (very likely), mean=3.25, SD=1.45, n=28 groups and 82 students.

of all the aspects that comprehensive feedback could encompass.

- *"The tool highlights some areas of improvements but was not able to accurately provide all errors and suggestions. The feedback obtained by mentors and the professor during the demo was much more insightful."*

The ratings for Q5 (Replaceability), as depicted in Figure 5.5, show that students have varied perspectives on whether the generated feedback can replace the instructor feedback. Some students (50% of them rated 4 or above out of 5) believe it can, while others feel that instructors can provide more helpful and nuanced insights.

- *"The feedback puts light on the tests not being described in detail, I think it was a good point but the particulars that we get from peers, and professor are better where there is someone actually monitoring your work."*

5.3.2 Interview Summary

The following key points were distilled from the semi-structured interview with the instructor. First, the instructor expressed cautious optimism regarding the LLM-generated feedback. While acknowledging its potential to assist students in improving project reports, the instructor highlighted the need for feedback to be faithful and more specific. Second, the instructor articulated an expectation for the flexibility to adjust various aspects of the generated feedback, such as its tone and areas of focus, to better align with instructional objectives. Lastly, regarding the integration of automated feedback systems into existing pedagogical frameworks, the instructor considered that directly sending the generated feedback to students still poses risks. Thus, the instructor deemed the most pragmatic approach to be using the LLM-generated feedback as the initial feedback draft.

5.4 Analysis and Discussion

5.4.1 Insight 1: Faithfulness is a major concern

The first insight from our analysis is the concern regarding faithfulness. Despite the LLM-generated feedback closely resembling instructor feedback in fluency and coherence, it may contain hallucinated content that is erroneous, misleading, or entirely irrelevant to the original project reports. Both the students and the instructor in this study expressed apprehension that such hallucinated content could confuse or mislead students, and consider it to significantly impact the reliability of the automated feedback system. To address the concern of hallucination, future research could attempt to understand the underlying causes of hallucination (Xu et al. 2024), and explore various hallucination detection and post-editing techniques (Maynez et al. 2020). These efforts can pave the way for the deployment of automated feedback systems in real classrooms.

5.4.2 Insight 2: Project-specific feedback is expected

The second insight underscores that the LLM-generated feedback tends to be generic. The students expressed a strong preference for feedback that is tailored to their project reports, as they felt that specific and detailed feedback would be more helpful for their learning. This sentiment was echoed by the instructor during the interview, further emphasizing the significance of project specificity. We speculate that the lack of specificity is primarily due to the feedback-generation system itself lacking relevant knowledge to provide detailed feedback. We also observed from preliminary experiments that forcing an increase in specificity appears to decrease its faithfulness. To this end, future research could explore techniques such as retrieval-augmented generation (RAG) (Lewis et al. 2020c) to incorporate the required knowledge for providing more project-specific feedback.

5.4.3 Insight 3: Human feedback remains irreplaceable

The third insight emphasizes the irreplaceable value of human feedback. While LLM-generated feedback offers valuable assistance, the students believe it cannot replace the nuanced understanding and personalized insights provided by human instructors. The instructor also felt that the feedback did not always align with the pedagogical objectives. Therefore, while automated feedback systems can supplement and enhance the learning process, they currently cannot entirely replace the role of human instructors. Instead, a synergistic approach may be adopted, where automated feedback systems support and complement human feedback, maximizing the benefits of both approaches. For example, the instructor found that using automated feedback as an anchor when personally crafting feedback can considerably reduce the time and effort needed.

5.4.4 Insight 4: Interactive feedback could be the way forward

The last insight is that interactive feedback can be a promising direction for future improvements. In the survey, some students expressed a desire for the feedback system to allow them to inquire about specific aspects of the feedback that they find confusing or in need of further explanation. Existing static feedback, although valuable, lacks the flexibility to address individual concerns or queries. In contrast, interactive feedback systems offer the potential to engage students more actively in the feedback process by providing them with opportunities to seek clarification and additional information as needed, which may enhance the personalization and effectiveness of feedback. Thus, the focus of future research may shift from static feedback systems to interactive feedback systems that enable dynamic student engagement and foster a more effective learning experience.

5.5 Conclusion, Limitations, and Future Research

LLM-based automated feedback systems hold immense potential for transforming the educational landscape (Roschelle et al. 2020). However, the absence of perspectives from students and instructors leaves uncertainties regarding their effectiveness and limitations in practical educational settings. To bridge this gap, we deployed Insta-Reviewer in a computer science course. We then solicited student opinions on generated feedback through a questionnaire and conducted a semi-structured interview with the instructor. The results demonstrate the value of the feedback, but also reveal areas requiring enhancement, such as faithfulness and specificity. This study contributes insights towards improving feedback-generation systems to better fulfill the needs of students and instructors.

Limitations and future work: There are two main limitations to this study. Firstly, the study was conducted in a single graduate-level computer science course at one university. This limits the generalizability of the findings to other educational contexts, particularly those with different subject matter, student populations, or instructional approaches. Therefore, future research can be conducted in different environments. Second, this study relies on anonymous questionnaire surveys and semi-structured interviews for collecting opinions. While these methods provide valuable insights, they may not capture the entire spectrum of perspectives or experiences, and the findings may be subject to bias or limitations inherent in self-reporting. Consequently, future studies could consider complementing these methods with other quantitative measures or extrinsic evaluations.

Part II

Machine-learned Evaluation of System-generated Feedback

CHAPTER

6

BERT-based Multi-task Learning Models for Evaluating Feedback

6.1 Introduction

Feedback is an essential component of the educational process because it helps students bridge the gap between their actual and desired performance by giving them clear guidance on how to enhance their learning (Hattie and Timperley 2007; Alharbi 2017; Kusairi 2020; Bulut and Wongvorachan 2022). However, providing feedback is a highly repetitious process that requires a substantial amount of time and effort on the part of the instructors (Winstone et al. 2017; Henry et al. 2020). Therefore, with the advancement of artificial intelligence, there has been a surge of interest in developing feedback-generation models to produce textual feedback on student work automatically. For example, Lu and Cutumisu (2021) has presented a unified framework for English essay scoring and feedback generation. In Chapter 4, we have proposed a large language model (LLM)-based feedback-generation system, dubbed Insta-Reviewer, for generating feedback on students' project reports.

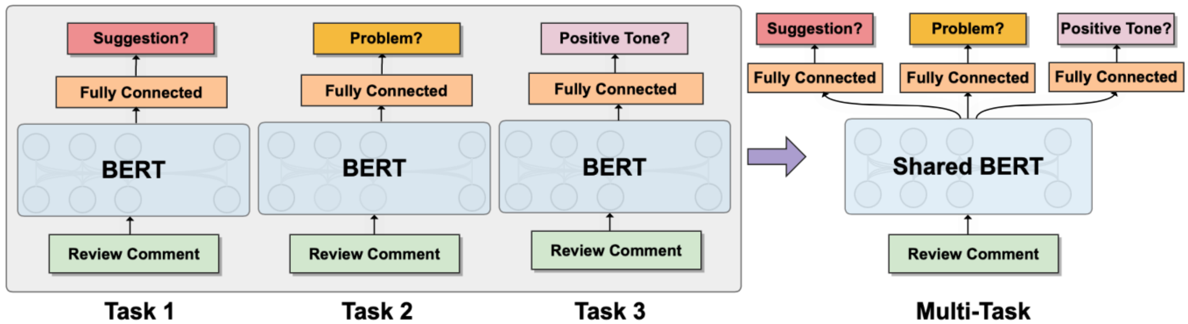

Figure 6.1: Illustration of the single-task learning (STL) and multi-task learning (MTL) settings. STL treats the problem as three independent tasks. MTL jointly learns three tasks.

Nevertheless, the evaluation of system-generated feedback remains largely unresolved, and there is no consensus on the criteria for assessing the generated feedback, primarily because textual-feedback generation is a highly open-ended task. A feedback-generation system can produce multiple reasonable and valuable reviews for a given student's work, but these reviews can be vastly different. Therefore, neither the content-overlapping metrics (measuring the word or n-gram overlap between texts), such as BLEU (Papineni et al. 2002) and ROUGE (Lin 2004), nor the embedding-based metrics (measuring the semantic similarity between texts), such as BERTScore (Zhang et al. 2020), are capable of capturing the quality of system-generated feedback. For this reason, human evaluation, which requires human evaluators to judge the quality of system-generated text along some specific dimensions (e.g., tone), remains the gold standard when assessing feedback quality.

However, human evaluation is time-consuming and costly, and the heavy annotation effort can result in potential inconsistencies in evaluation quality, which may prevent researchers from comparing results across methods or lead to erroneous conclusions (Celikyilmaz et al. 2020). Thus, one way to alleviate this problem is to build machine-learning models that are trained on a relatively small number of human-evaluation labels, to mimic human evaluators in judging different quality dimensions of feedback. For example, Zingle et al. (2019) utilized various rule-based, machine-learning, and deep-learning methods to detect whether a feedback text contains suggestions. Xiao et al. (2020) leveraged different neural networks to determine whether a feedback message contains problem statements. Nevertheless, previous efforts to automatically evaluate feedback have focused on building models that assess a single quality metric (as illustrated in the left part of Figure 6.1).

To our knowledge, no attempt has been made to investigate a multi-task learning (MTL) model for judging multiple quality dimensions simultaneously (as shown in the right part of Figure 6.1). There are two motivations for using MTL models: 1) the problem naturally leads itself well to MTL, as multiple quality dimensions are typically required for a comprehensive and accurate evaluation. If we treat this MTL problem as multiple independent single tasks, the development cost and total model size will increase linearly with the number of dimensions utilized for evaluation, instead of remaining constant; 2) MTL can improve data efficiency and generalizability, which implies that jointly learning the tasks can lead to performance improvement compared to learning them individually (Crawshaw 2020). Specifically, MTL can be viewed as a form of inductive transfer learning that can help improve the performance of each task by introducing inductive bias (Caruana 1997).

In this chapter, we propose to use a BERT-based MTL model to evaluate feedback quality along multiple quality dimensions simultaneously. For the backbone model, we employ BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2019), a pre-trained large language model (LLM). BERT is a transformer model (Vaswani et al. 2017) pre-trained on a large corpus in a self-supervised fashion, which can significantly reduce the need for labeled data. We first compare a BERT-based single-task learning (STL) model with the previous GloVe-based STL model (Pennington et al. 2014). We then set up the BERT-based MTL model to concurrently learn multiple tasks. Experimental results demonstrate that the BERT-based STL model can significantly outperform the previous GloVe-based STL model by at least 8% in F_1 score on tasks of judging a single quality metric. MTL can further improve average performance and reduce development and maintenance expenses.

Our main contributions are: 1) we propose a BERT-based STL model for mimicking human evaluators in judging the quality of feedback in different dimensions; 2) we further utilize a BERT-based MTL model to simultaneously assess multiple quality dimensions of feedback; 3) we also collect two new datasets to facilitate future research endeavors.

The rest of the chapter is organized as follows: Section 6.2 elaborates on the GloVe-based STL model and our proposed BERT-based STL and MTL models. Section 6.3 details our experimental setup and explains the evaluation metrics. Section 6.4 presents and discusses our experimental results on two feedback datasets. Section 6.5 concludes this chapter, mentions the limitations of our research, and provides discussion about future work.

6.2 Methodology

In this section, we detail our machine-learning models for mimicking human evaluators to judge the quality of feedback in different dimensions. We first formally define the problem in Section 6.2.1. Then, we present the GloVe-based STL baseline model in Section 6.2.2. After that, we briefly introduce our backbone model BERT in Section 6.2.3. Finally, we elaborate on the implementation of BERT-based STL and MTL models in Section 6.2.4.

6.2.1 Problem Formulation

We formulate the task of judging the quality of feedback in different dimensions as a text classification problem. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote the feedback space or a collection of n feedback messages, and x_i denotes a feedback instance. The label set $L = \{l^1, l^2, \ldots, l^m\}$ contains m different labels (e.g., *Suggestion*, *Problem*, *Positive tone*) for each review instance. The labels associated with each review x_i form a subset of L, which can be represented as an m-dimensional binary label vector $y_i = [y_i^1, y_i^2, \ldots, y_i^m]$ where $y_i^j \in \{0, 1\}$, with $y_i^j = 1$ indicating that l^j is a proper label for the review x_i, and "0" indicating otherwise. Then, we have a set $Y = \{y_1, y_2, \ldots, y_n\}$, where $y_i \in \{0, 1\}^{1\times m}$ is the label vector for the sample x_i,

Then, the problem can be formally described as:

$$\mathscr{C} : X \mapsto Y \tag{6.1}$$

where the model, or classification function, $\mathscr{C}$ takes a feedback message $x \in X$ as the input, and produces a label or labels for the given feedback x. For the STL setting, we craft m different classifiers $\mathscr{C}_{STL} = \{\mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_m\}$, where $\mathscr{C}_j$ is the classification function that maps from $x \in X$ to $y^j \in \{0, 1\}$. For the MTL setting, instead of learning m classification functions, we build only one classification function $\mathscr{C}_{MTL}$ that maps from $x \in X$ to $y \in \{0, 1\}^{1\times m}$.

The objective of the task is to effectively model the classification function $\mathscr{C}_{MTL}$ using the training dataset, so that it can produce accurate labels for unseen feedback. In this work, the classification function $\mathscr{C}_{MTL}$ is crafted based on BERT (Devlin et al. 2019). Specifically, we will first demonstrate that the BERT-based STL model is more effective than the previous GloVe-based model in constructing the STL classification functions $\mathscr{C}_{STL} = \{\mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_m\}$. Then, we will show that $\mathscr{C}_{MTL}$ can be built using a BERT-based MTL model, which is more efficient and effective than using the BERT-based STL models for feedback evaluation.

6.2.2 Global Vectors for Word Representation (GloVe)

We first implemented a baseline STL model to craft $\mathscr{C}_{MTL}$ by leveraging pre-trained GloVe (Global Vectors for Word Representation)[1] (Pennington et al. 2014) word embeddings. GloVe is the leading method used in previous studies on judging the quality of feedback (e.g., Zingle et al. (2019); Xiao et al. (2020)). The word embeddings (i.e., word vectors) of GloVe are trained such that their dot product equals the logarithm of the words' probability of co-occurrence. To build a GloVe-based STL model, we added a BatchNormalization layer (Ioffe and Szegedy 2015) on top of GloVe, and the aggregate representation of the input sequence for classification was obtained by average-pooling the output of the BatchNormalization layer. Then, a dense layer with the ELU activation function (Clevert et al. 2015) and a dropout layer (Srivastava et al. 2014) with a 0.1 dropout rate were stacked. Finally, a dense layer with the sigmoid activation function was stacked on the top for performing classification.

6.2.3 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a state-of-the-art pre-trained LLM proposed by Devlin et al. (2019). It has advanced the state-of-the-art results in various NLP tasks and significantly reduced the need for labeled data. Each BERT model consists of 12 encoder blocks of the Transformer model (introduced in Section 2.2.2) and pre-trained on large corpora crawled from the Internet over two self-supervised pre-training tasks (viz., masked language modeling (MLM) and next sentence prediction (NSP)). The input representation is constructed by summing the corresponding token and positional embeddings. The length of the output sequence of BERT is the same as the input length, and each input token has a corresponding representation in the output. Following previous research (e.g., Sun et al. (2019)), we employed the output representation of the first token "[CLS]" (a special token added to the sequence) as the vector representation of the input sequence for downstream classification tasks.

Specifically, the BERT framework consists of two steps: *pre-training* and *fine-tuning*, where the pre-training stage is significantly more computationally expensive. During pre-training, the model is trained on unlabeled data, BooksCorpus (800M words) (Zhu et al. 2015) and English Wikipedia (2,500M words), over the two pre-training tasks to acquire some "knowledge" of language (e.g., grammar). The pre-training step produces a foundation model that can be trained (i.e., fine-tuned) much more efficiently on the downstream tasks

[1] glove.42B.300d checkpoint: `https://nlp.stanford.edu/projects/glove/`

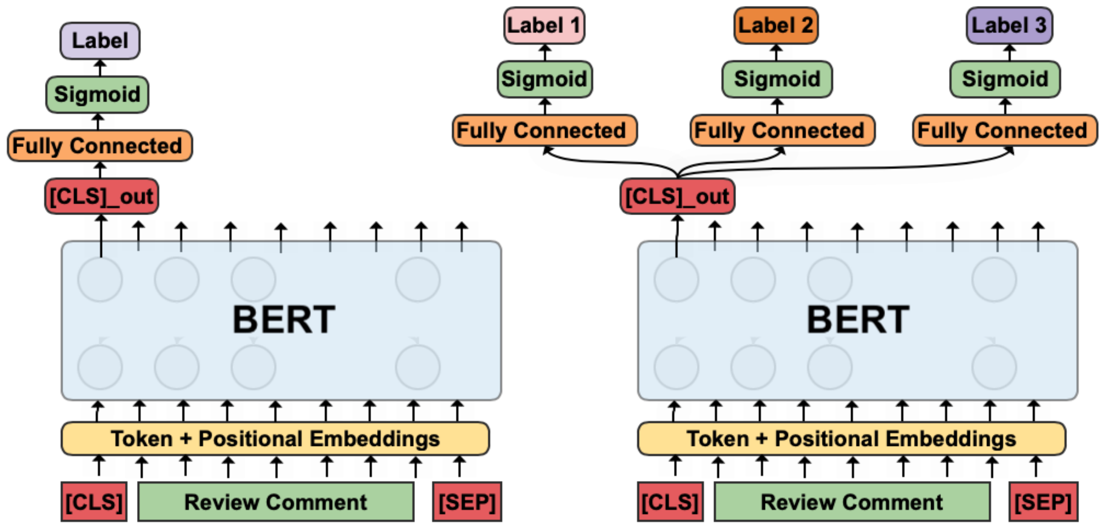

Figure 6.2: The architectures of the BERT-based STL (left) and MTL (right) models.

with much less labeled data. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and then all of the parameters are fine-tuned using labeled data from the downstream tasks (e.g., text classification). For this study, we use the HuggingFace pre-trained bert-based-uncased checkpoint [2] to initialize the models and then fine-tune the models with annotated feedback messages for evaluating the quality of feedback.

6.2.4 BERT-based Single-task and Multi-task learning Models

As mentioned in the BERT paper (Devlin et al. 2019) and other studies (e..g, Sun et al. (2019)), the pre-trained BERT model can be fine-tuned with just one additional task-specific output layer to create state-of-the-art models for a wide range of tasks, including text classification. In addition, our preliminary results showed that stacking additional layers (e.g., dense layers and dropout layers) provides no gain for the results. Thus, for the BERT-based STL model, only one dense layer with the sigmoid function was added on top of the BERT model, as shown in Figure 6.2. For building the BERT-based MTL model, we utilized a shared BERT, which allows information flow between a set of related tasks, to learn the common ideas between them. Then, we stacked three separate dense layers with the sigmoid functions on the top of the shared BERT representation, one for each task (each quality dimension).

[2]bert-base-uncased checkpoint: `https://huggingface.co/bert-base-uncased`

6.3 Experimental Setting and Evaluation

In this section, we first describe the input preparation in Section 6.3.1. Then, we introduce the experimental setup in Section 6.3.2, and present our evaluation metrics in Section 6.3.3.

6.3.1 Input Preparation

Text Pre-processing: For the system-generated feedback (SGF) dataset, we simply lowercased all feedback and added two special tokens (viz., [CLS], [SEP]) as required for BERT. The [CLS] token was added to the beginning of each review. The [SEP] token was added at the end of each review. For the peer-review feedback (PRF) dataset, we pre-processed the feedback according to the following steps. First, URL links in the feedback were removed using a regular expression. Then, we lowercased all feedback and leveraged a spellchecker API[3] to correct all typos and misspellings. Finally, the two special tokens were added.

Subword Tokenization: The tokenizer used by BERT is a subword tokenizer called "WordPiece" with a 30,522 token vocabulary (Wu et al. 2016). Traditional word tokenizers suffer the out-of-vocabulary (OOV) word problem. However, a subword tokenizer can alleviate the OOV problem. It splits a text into subwords and then converts them to token IDs.

Input Representation: The token IDs of each sequence were padded or truncated to 100 and then passed through a trainable embedding layer to be converted to 768-dimensional token embeddings. The input representation for BERT was constructed by summing the token embeddings and the positional embeddings that encode order dependencies.

6.3.2 Experimental Setup

Loss Functions: As we mentioned, we formulated the problem as a text classification task. Therefore, we utilized the cross-entropy objective function to optimize the models. Specifically, for the BERT-based STL models, we calculated the cross-entropy loss. For the BERT-base MTL models, the cross-entropy loss was utilized for each task. The total loss of the MTL model was calculated by summing up the cross-entropy losses of each task.

[3] pyspellchecker API: `https://pypi.org/project/pyspellchecker/`

Cost-Sensitive method: A cost-sensitive method (Elkan 2001) was used for alleviating the problem of class imbalance. Concretely, we strongly penalized mistakes on the minority class by weighting the cross-entropy loss function during training based on the frequency of each class in the training dataset, to improve generalization on the minority class.

Hyperparameters: We used the HuggingFace pre-trained BERT checkpoint to initialize our models that implemented using Tensorflow. The hidden size for BERT is 768. We then fine-tuned the BERT-based STL and MTL models with a batch size of 8, a max sequence length of 100, a learning rate of 2e-5/3e-5, epochs of 10 with the early-stopping method (Prechelt 2002), and the Adam optimizer (Kingma and Ba 2014) with β_1=0.9, β_2=0.99.

Hardware Setup: We implemented the models using Tensorflow and Huggingface packages. We trained the BERT-based STL and MTL models on an NVIDIA RTX5000 GPU (16GB).

6.3.3 Evaluation Metrics

We used Cohen's kappa coefficient (κ) (Cohen 1960) and macro-F_1 score (F_1) to evaluate models. Cohen's kappa coefficient is a robust statistic that is utilized to assess the agreement between two raters, which can therefore measure the agreement between the human evaluator (i.e., ground-truth labels) and the machine-learned evaluator (i.e., model predictions). As suggested by Cohen (1960), the kappa score can be interpreted as follows: values ≤ 0 means no agreement, 1–20 as none to slight, 21–40 as fair, 41–60 as moderate, 61–80 as substantial, and 81-100 as almost perfect agreement. Moreover, the macro-F_1 score is the unweighted mean of F_1 scores calculated per class, so there is no distinction between highly and poorly populated classes. Thus, the macro-F_1 score provides a better picture of whether the model performs well on all classes than the micro-F_1 score (Grandini et al. 2020).

6.4 Experimental Results and Discussion

In this section, we present our experimental results and provide some discussion. Section 6.4.1 compares the GloVe-based STL model and the proposed BERT-based STL model. Then, Section 6.4.2 compares the BERT-based MTL model and the BERT-based STL model. Finally, Section 6.4.3 investigates other advantages of employing the BERT-based MTL model.

Table 6.1: Performance evaluation of GloVe (baseline), Single-task BERT (STL-BERT) and Multi-task BERT (MTL-BERT) on the *System-generated Feedback (SGF)* dataset (input length limit is 100 subword tokens). The best results in each setting are marked in bold. CI = 95%.

	Suggestion		**Problem**		**Pos. Tone**		**Average**	
	F_1	κ	F_1	κ	F_1	κ	F_1	κ
Training with 150 labeled data samples								
1 STL-GloVe	84.05±9.8	68.36±19.3	73.14±6.1	47.69±11.5	80.47±6.7	61.35±13.2	79.22	59.13
2 STL-BERT	**97.36**±1.44	**94.73**±2.9	94.33±3.4	88.67±6.7	94.02±5.9	88.09±11.9	95.24	90.50
3 MTL-BERT	97.04±2.2	94.07±4.4	**95.07**±4.2	**90.15**±8.4	**95.40**±4.1	**90.83**±8.1	**95.84**	**91.69**

6.4.1 Study 1: Comparison between GloVe and BERT

The purpose of our first experiment is to demonstrate the necessity of using the advanced LLM BERT, by comparing the BERT-based STL model (STL-BERT) with the previous GloVe-based STL model (STL-GloVe). We tested the models on the system-generated feedback (SGF) dataset (introduced in Section 3.2) and the peer-review feedback (PRF) dataset (introduced in Section 3.3). The latter contains more syntactically diverse feedback, which allows us to test the prospects of the models. Besides, we evaluated the feedback on three quality dimensions (viz., containing suggestions, containing problems, and adopting positive tone). The evaluation results of the two STL models are shown in Table 6.1 and Table 6.2.

For the SGF dataset (Table 6.1), STL-BERT can outperform STL-GloVe by a large margin in terms of both F_1 scores and Cohen's kappa scores. STL-GloVe can achieve substantial agreement ($60 < \kappa < 80$) with human judgment on "Suggestion" ($\kappa = 68.36$) and "Positive Tone" ($\kappa = 61.35$), but it can only achieve moderate agreement ($40 < \kappa < 60$) on "Problem" ($\kappa = 47.69$). However, STL-BERT can achieve an almost perfect level of agreement ($\kappa > 80$) with human judgment on all quality dimensions of feedback, with κ scores of 94.73, 88.67, and 88.09 on "Suggestion," "Problem," and "Positive Tone," respectively. Therefore, the results show that STL-BERT considerably outperforms STL-GloVe on the SGF dataset.

For the PRF dataset (Table 6.2), STL-BERT can also significantly outperform STL-GloVe in different settings. When training the models with 1000, 2000, and 3000 labeled data samples, STL-BERT can outperform STL-GloVe with gaps of 21.53, 18.04, and 17.7 in the average kappa scores, respectively. Specifically, when training with 3000 samples, STL-GloVe can achieve moderate agreement with human judgment on average ($\kappa = 59.79$). However, STL-BERT can achieve substantial agreement on average ($\kappa = 77.49$). In addition, we can

Table 6.2: Performance evaluation of GloVe (baseline), Single-task BERT (STL-BERT) and Multi-task BERT (MTL-BERT) on the *Peer-review Feedback (PRF)* dataset (input length limit is 100 subword tokens). The best results in each setting are marked in bold. CI = 95%.

	Suggestion		**Problem**		**Pos. Tone**		**Average**	
	F_1	κ	F_1	κ	F_1	κ	F_1	κ
Training with 1000 labeled data samples								
1 STL-GloVe	78.68±1.2	57.67±2.3	79.24±1.2	58.50±2.5	70.13±2.3	41.91±4.3	76.02	52.69
2 STL-BERT	88.6±1.1	77.32±2.1	**89.68**±1.3	**79.39**±2.6	82.86±1.9	65.96±3.7	87.05	74.22
3 MTL-BERT	**90.53**±1.9	**81.1**±3.8	88.86±1.3	77.72±2.5	**83.45**±2.5	**66.95**±4.9	**87.61**	**75.25**
Training with 2000 labeled data samples								
4 STL-GloVe	81.91±1.1	63.96±2.1	80.27±1.7	60.56±3.4	73.79±2.2	48.39±4.1	78.66	57.64
5 STL-BERT	90.95±1.9	81.92±3.9	**89.67**±0.5	**79.34**±1.1	82.73±1.2	65.78±2.4	87.78	75.68
6 MTL-BERT	**91.85**±1.4	**83.72**±2.8	89.42±0.9	78.83±1.8	**84.98**±1.5	**70.01**±3.0	**88.75**	**77.52**
Training with 3000 labeled data samples								
7 STL-GloVe	83.29±0.7	66.66±1.3	81.79±1.5	63.59±3.0	74.12±1.8	49.13±3.4	79.73	59.79
8 STL-BERT	**91.86**±1.1	**83.72**±2.3	**89.96**±1.3	79.93±2.6	84.32±1.4	68.82±2.9	**88.71**	77.49
9 MTL-BERT	91.26±1.0	82.53±2.1	89.70±1.3	**80.05**±3.2	**84.97**±1.2	**70.02**±2.3	88.64	**77.53**

find similar results when training with 1000 and 2000 samples. Thus, experimental results demonstrate that STL-BERT appreciably outperforms STL-GLoVe on the PRF dataset.

In summary, the experimental results clearly demonstrate that the BERT-based STL model yields a substantial improvement over the previous GloVe-based method, and support the necessity of using the advanced BERT model. The machine-learned evaluator (i.e., STL-BERT) can achieve an average inter-rater agreement (κ) of 90.5 with the human evaluator on the system-generated feedback, which is considered to be almost perfect agreement. Furthermore, despite the fact that the peer-review feedback is significantly more syntactically diverse, the machine-learned evaluator achieves an average inter-rater agreement (κ) of 77.49, which approaches the threshold for almost perfect agreement.

6.4.2 Study 2: Performance of MTL-BERT

The goal of the second experiment is to find out whether we can use MTL to evaluate multiple quality dimensions of feedback simultaneously. To this end, we compared the performance of the BERT-based MTL model with that of the BERT-based STL model. We tested the models on both the system-generated feedback (SGF) dataset and the peer-review feedback (PRF) dataset. In each setting, we trained and tested the models on the same sets of data. The feedback is evaluated on the three quality dimensions (viz., containing suggestions, detecting problems, and adopting positive tone). The evaluation results are displayed in Table 6.1 (in rows 2 and 3) and Table 6.2 (in rows 2 and 3, 5 and 6, 8 and 9).

For the SGF dataset (Table 6.1), both MTL-BERT and STL-BERT can achieve almost-perfect agreement with the human evaluator on all quality dimensions. MTL-BERT can outperform STL-BERT by 0.6 on the average F_1 score and 1.2 on the average κ score. Concretely, MTL-BERT can outperform STL-BERT by 1.48 and 2.74 in terms of κ scores on "Problem" and "Positive Tone." However, on "Suggestion," the F_1 score and κ score of MTL-BERT are slightly lower than those of STL-BERT with differences of 0.32 and 0.66, respectively. Overall, the experimental results demonstrate that MTL can further improve the average performance on the SGF dataset. However, not all three tasks benefit from using MTL.

For the PRF dataset (Table 6.2), MTL-BERT can outperform STL-BERT with gaps of 1.0 and 1.8 on average κ scores, when training with 1k and 2k samples, respectively. However, when training with 3k samples, they have similar scores. Specifically, when we have relatively limited training data (i.e., training with 1k and 2k samples), MTL-BERT can outperform STL-BERT on "Suggestion" and "Positive Tone." However, MTL-BERT performs worse on "Problem." When training with 3k samples, MTL-BERT performs better on "Positive Tone", similarly on "Problem," but worse on "Suggestion." Thus, the results imply that MTL has superior average performance on the PRF dataset, but not all tasks can benefit from it.

In conclusion, by comparing MTL-BERT with STL-BERT, the second study suggests that jointly learning related tasks can improve the average performance of feedback evaluation. MTL-BERT is a superior choice over STL-BERT for implementing the machine-learned evaluator. The results on the PRF dataset also indicate that MTL appears to increase data efficiency when the number of training samples is relatively limited. However, the results also show that MTL is not advantageous for all tasks. This may be because one task may dominate the learning, resulting in inferior performance on other tasks, or the tasks need to be learned at different learning rates on account of discrepancies in their difficulty.

6.4.3 Study 3: Advantages of using MTL-BERT

In addition to the superior average performance, there are two other advantages of using the BERT-based MTL model. Firstly, MTL can significantly reduce development costs. Training only one MTL model eliminates the need to search for hyperparameters for multiple STL models. Secondly, MTL can considerably reduce maintenance expenses. Instead of deploying multiple BERT-based STL models, we maintain only one MTL model, which leads to reduced costs. Three BERT-based STL models have more than 328M parameters. However, if we employ the BERT-based MTL model, the number of parameters is reduced to 109M. This implies using MTL can considerably lower the hardware requirements.

6.5 Conclusion, Limitations, and Future Research

How to efficiently evaluate generated feedback remains an open question, and human evaluation is still considered the gold standard when assessing feedback. However, human evaluation is time-consuming, costly, and prone to inconsistency. In this chapter, we present BERT-based models that are trained on a relatively small number of human-evaluation labels, to mimic human evaluators in judging the quality of feedback across different dimensions. Overall, the results demonstrate that the BERT-based STL model significantly outperforms the previous GloVe-based STL model. The BERT-based MTL model can further improve average performance and reduce development and maintenance expenses.

This study has three major limitations. Firstly, we evaluated the feedback on three quality dimensions. However, it is still unclear how MTL will perform if it learns more tasks simultaneously. Secondly, we focused mainly on the hard-parameter sharing approach for constructing MTL models. However, some studies have found that the soft-parameter sharing approach (e.g., Pahari and Shimada (2022)) might be a more effective way to craft MTL models. Thirdly, we assumed the dataset is "complete" (i.e., each sample has complete annotations on all dimensions). However, in practice, such as owning to the need to filter out unreliable annotations, certain samples might lack labels in specific dimensions.

In the future, we will focus on two directions for improving feedback-evaluation models. Firstly, while we have reduced the dependence on annotations by adopting a BERT-based multi-task learning model, the acquisition of these labels remains a laborious process. Hence, it is worthwhile to explore how to further diminish the reliance of feedback-evaluation models on annotations, with the ultimate goal of utilizing zero labels. Furthermore, we can delve into the prospect of transferring the embedded "knowledge" from a feedback-evaluation model trained for one domain to another domain. The "transfer" can facilitate a more efficient creation of feedback-evaluation models for other domains.

CHAPTER

7

An Incremental Zero-shot Learning Approach for Assessing Feedback

7.1 Introduction

Feedback is a fundamental aspect of teaching and learning, as it provides explicit guidance to help students progress toward their educational objectives (Hattie and Timperley 2007; Alharbi 2017; Kusairi 2020; Bulut and Wongvorachan 2022). The need to deliver feedback promptly and at scale has sparked a surge of interest in developing automated feedback-generation systems (Deeva et al. 2021; Dai et al. 2023). Nevertheless, one significant hurdle in developing such systems lies in the absence of efficient techniques for evaluating the generated feedback. Efficient evaluation methods are indispensable for hyper-parameter tuning and model comparison. However, existing automatic metrics, which gauge the generation by measuring the overlap or semantic similarity between texts, still fall short in replicating human judgments (Reiter and Belz 2009). Thus, feedback generation still requires human evaluation to provide unparalleled insight into model performance.

Human evaluation requires evaluators to assess the credibility of each feedback message in accordance with specific quality dimensions (e.g., containing suggestion statements). However, this process is both costly and time-consuming. More critically, the laborious annotation effort can result in inconsistencies in evaluation quality and may lead to erroneous conclusions (Celikyilmaz et al. 2020). Hence, researchers have endeavored to automate the evaluation of feedback quality by employing machine-learning models. These models are trained on a relatively small number of human-evaluation labels, to emulate human evaluators to judge the quality of feedback across different dimensions. For instance, Xiao et al. (2020) utilized various neural networks to determine whether the feedback contains problem statements. Chapter 6 also built a multi-task learning model to decide whether the feedback contains suggestion and problem statements, and adopts positive tone.

There is no doubt that utilizing such feedback-evaluation models necessitates significantly less human effort compared to manual evaluation. However, even with the possibility of collecting the annotations incrementally by crowd-sourcing students' labels each term as a cost-saving measure, obtaining sufficient annotations for training these evaluation models remains an onerous undertaking. Therefore, it is worthwhile to explore methods that can further alleviate the reliance on annotations, ideally using zero annotations. Specifically, the learning process of feedback-evaluation models can be delineated into two phases: a *"cold-start" phase* (i.e., devoid of historical data) and an *"incremental learning"* phase (i.e., with incrementally collected labels). Accordingly, the objective is to develop a feedback-evaluation model that can achieve as good as possible performance in the "cold-start" phase, and surpass previous methods after training on incrementally collected data.

In this chapter, we present an approach, named *Incremental Zero-shot Learning* (IZSL), for mitigating the dependence on annotations for developing feedback-evaluation models. The core idea of the method is to treat the problem of evaluating feedback quality as a natural language inference (NLI) task and utilize the pre-trained BART-based NLI model (Yin et al. 2019) to mimic human evaluators to judge different quality dimensions of feedback. Pre-training the BART model (Lewis et al. 2020a) on an NLI corpus named MultiNLI (Williams et al. 2018) allows it to perform feedback evaluation even without fine-tuning. Our results demonstrate that the IZSL model can achieve acceptable performance in the "cold-start" condition on different datasets, and IZSL can substantially outperform previous BERT-based feedback-evaluation methods (Devlin et al. 2019; Jia et al. 2021) after training on the same incrementally collected data, which is equivalent to reducing reliance on labels.

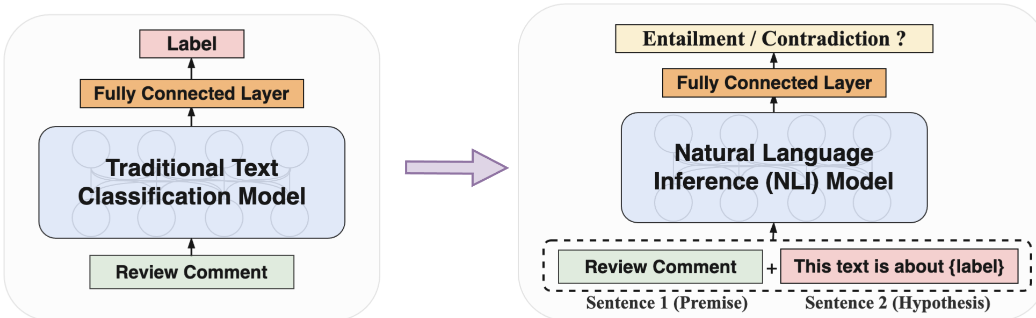

Figure 7.1: The key idea of IZSL is to convert the problem of evaluating peer-feedback comments into an NLI problem. The left part shows the traditional text classification setting for assessing feedback comments. The right part shows the NLI setting that treats the feedback message as the premise and uses the label to formulate the hypothesis.

Our main contributions are: 1) We identify the primary limitation of existing feedback-evaluation models and pioneer the exploration of using zero labels for feedback evaluation; 2) we propose a BART-based NLI model for mimicking human evaluators in judging the quality of feedback in different dimensions; 3) we highlight the potential of combining prompt engineering and LLMs as a promising direction for advancing this domain.

The rest of this chapter is organized as follows: Section 7.2 initially formulates the research problem, then elaborates on the proposed BART-based NLI (i.e., IZSL) model, and details our experimental setup. Section 7.3 showcases and discusses our experimental results on two feedback datasets. Lastly, Section 7.4 concludes this chapter, acknowledges the limitations of our research, and provides some discussions about future studies.

7.2 Methodology

In this section, we detail our IZSL approach for mimicking human evaluators to judge the quality of feedback in different dimensions. We first formally define the research problem in Section 7.2.1. Then, we present the BART-based NLI (i.e., IZSL) model in Section 7.2.2. After that, we briefly introduce the previous BERT-based model, which is employed as our baseline, in Section 7.2.3. Finally, we describe the experimental setup in Section 7.2.4.

7.2.1 Problem Formulation

We formulate the task of evaluating system-generated feedback as a text classification problem. Suppose that during the t-th semester of a class, we could collect a dataset $D^t = (X^t, Y^t)$ consisting of N^t data samples, where $X^t = \{x_1^t, x_2^t, \ldots, x_{N^t}^t\}$ denotes a set of N^t feedback messages annotated in the t-th semester, and Y^t denotes corresponding labels indicating whether the feedback provides suggestions, mentions problems, and adopts positive tone. In practice, as a cost-saving measure, these annotations can be obtained by, e.g., asking reviewees to determine if the received feedback contains those features. Specific ways for obtaining reliable labels fall within the scope of research on reputation systems and are therefore not discussed in this thesis. Additionally, it is worth noting that the labels Y^t can only be used for training after they are collected, i.e., after the t-th semester.

First, **in the "cold-start" condition** (i.e., without any historical data, in the 0-th semester), the task of IZSL is to craft a classifier $\mathscr{C}_{IZSL}$ that can effectively make predictions for system-generated feedback X^0 without leveraging any domain data to fine-tune the model. The ambitious objective of IZSL in this phase is to attain the best possible performance.

Then, **in the incremental learning phase** (i.e., $t > 0$), we would have historical data $D^{<t}$, where $D^{<t}$ means $(D^0, D^1, ..., D^{t-1})$ (the data we collected in the first $(t-1)$ semesters). The task of IZSL in this phase is to update the classifier $\mathscr{C}_{IZSL}$ using all historical data $D^{<t}$, and make more accurate predictions for the labels of system-generated feedback X^t.

7.2.2 Incremental Zero-shot Learning

We now describe our IZSL approach for classifying feedback messages. The overall idea of IZSL is to convert a text classification problem into a natural language inference (NLI) problem. As shown in Figure 7.1, traditional text classification models are trained to take a piece of text as input and output 0 or 1 to indicate the class to which the text belongs. The NLI task, on the other hand, is to determine whether, given a premise, a hypothesis is true (entailment) or false (contradiction). Specifically, we typically consider the text to be classified (i.e., a feedback message) as the premise, and construct the hypothesis using a prompt containing the class name of the label. For example, the hypothesis (i.e., prompt) could be "This text is about {label}," where "{label}" is either "suggestions" or "problems". Then, if the NLI model tells us that the premise is likely to entail the hypothesis, we can conclude that the label is associated with the input feedback message and vice versa.

In this work, we use BART (Lewis et al. 2020a) to craft the NLI model and initialize all parameters with the "bart-large-mnli" checkpoint[1] (Yin et al. 2019), which is pre-trained on the multi-genre NLI (MultiNLI) dataset (Williams et al. 2018). We attempt to construct the hypothesis using a variety of the most commonly used prompts (e.g., "This text is about problems," "This text contains suggestions," "This text is positive," "This text uses positive tone"), test them on a validation set, and pick the one with the lowest loss to accomplish our task. **In the "cold-start" condition**, using the pre-trained weights allows us to have an out-of-the-box NLI model for assessing feedback quality for any curriculum without needing historical data. This is impossible for traditional text classification models, since they need domain data to tune the output fully-connected layer. Then, **in the incremental learning phase**, we leverage incrementally collected data to further fine-tune the NLI model.

7.2.3 Baseline Classification Method

Although traditional text classification models cannot be applied in the "cold-start" condition, a BERT-based classifier is implemented to compare the performance of IZSL in the incremental learning phase. As introduced in Chapter 6, BERT is a pre-trained LLM proposed by Devlin et al. (2019) and can be fine-tuned with just one additional task-specific output layer to create state-of-the-art models for NLP tasks, such as text classification. Our preliminary results also indicated that stacking additional layers (e.g., dense layers and dropout layers) has essentially no gain on the results. Thus, we constructed the classifier by stacking a dense layer with the sigmoid function on top of BERT. The parameters of BERT are initialized using a pretained checkpoint,[2] and the weights of the dense layer are randomly initialized using the uniform distribution. To establish a fair comparison with IZSL, we fine-tune the model utilizing the exact same data as when fine-tuning IZSL.

7.2.4 Experimental Setup

Training and Optimization Details. We implemented the models using PyTorch and Huggingface packages. We trained our models on eight NVIDIA RTX6000 or A5000 GPUs (24GB each) with a total batch size of 8, a learning rate of 2e-5/3e-5/5e-5, epochs of 10 with the early stopping method (Prechelt 2002), and the Adam optimizer (Kingma and Ba 2014).

[1] bart-large-mnli checkpoint: `https://huggingface.co/facebook/bart-large-mnli`
[2] bert-base-uncased checkpoint: `https://huggingface.co/bert-base-uncased`

Handling the Imbalanced Datasets. A cost-sensitive approach (Elkan 2001) was employed to alleviate the class imbalance. Specifically, we weight the cross-entropy loss based on the frequency of each class in the training set, which enhances the model generalization.

Evaluation Metrics. As in Chapter 6, we employed Cohen's kappa coefficient (κ) (Cohen 1960) and macro-F_1 score (F_1) to assess the model's performance. Cohen's kappa coefficient measures the agreement between two raters, such as the human evaluator (i.e., ground-truth labels) and the machine-learned evaluator (i.e., model predictions). Furthermore, the macro-F_1 score calculates the F_1 score for each class separately and then takes the unweighted average, which is more adequate for imbalanced datasets than micro-F_1.

7.3 Experimental Results and Discussion

In this section, we present our experimental results and provide a subsequent discussion. The ISZL model was tested on both the system-generated feedback (SGF) dataset and the peer-review feedback (PRF) dataset. We thoroughly review the outcomes to gain valuable insights into the effectiveness and potential limitations of IZSL. Section 7.3.1 delves into the performance of IZSL in the "cold-start" phase. Subsequently, Section 7.3.2 compares the performance of BERT and IZSL after training on the incrementally collected data.

7.3.1 Study 1: Performance in the cold-start phase

The objective of the first study is to examine the effectiveness of the IZSL model in the "cold-start" phase. This allows us to understand the performance that the model can achieve without using any labels (i.e., zero-shot). The evaluation results are displayed in the first rows (denoted as "Training with 0 labeled data samples") of Table 7.1 and Table 7.2.

For the SGF dataset (Table 7.1), IZSL demonstrates acceptable "cold-start" performance. However, it is worth noting that the performance of the IZSL model varies on different quality dimensions. Specifically, IZSL achieves a moderate level of agreement ($41 < \kappa < 60$) with human judgment on "Suggestion" ($\kappa = 42.04$) and near a moderate level of agreement on "Positive Tone" ($\kappa = 39.52$), but it can only attain a fair level of agreement ($21 < \kappa < 40$) on "Problem" ($\kappa = 26.09$). Despite the significant gap between the average κ of the IZSL model ($= 35.9$) and the average κ between the human annotators ($= 92.0$), considering that the model does not use any labeled data, this "cold-start" performance is acceptable.

Table 7.1: Performance evaluation of BERT (baseline) and IZSL on the *System-generated Feedback (SGF)* dataset. The best results in each setting are marked in bold. CI = 95%.

	Suggestion		Problem		Pos. Tone		Average	
	F_1	κ	F_1	κ	F_1	κ	F_1	κ
Training with 0 labeled data samples (zero-shot)								
1 IZSL	**69.89**±2.9	**42.04**±5.2	**62.34**±1.3	**26.09**±2.6	**67.14**±1.3	**39.52**±1.9	**66.46**	**35.88**
Training with 50 labeled data samples								
2 BERT	93.79±1.9	87.60±3.8	72.89±4.6	46.16±8.9	84.68±5.8	69.61±11.1	83.79	67.79
3 IZSL	**96.21**±0.7	**92.42**±1.4	**88.33**±4.5	**76.82**±8.9	**94.34**±0.6	**88.69**±1.2	**92.96**	**85.98**
Training with 100 labeled data samples								
4 BERT	95.99±1.7	91.99±3.4	81.61±7.6	63.38±15.2	94.03±3.5	88.09±6.9	90.54	81.16
5 IZSL	**96.49**±1.2	**93.00**±2.5	**89.31**±4.8	**78.71**±9.6	**94.69**±2.4	**89.39**±4.9	**93.50**	**87.03**
Training with 150 labeled data samples								
6 BERT	97.59±2.5	95.19±5.1	94.66±3.2	89.34±6.3	94.58±5.6	89.17±11.2	95.61	91.23
7 IZSL	**98.18**±0.9	**96.36**±1.8	**95.39**±3.2	**91.12**±6.2	**96.14**±2.4	**92.28**±4.9	**96.57**	**93.25**

For the PRF dataset (Table 7.2), the performance of IZSL (average $\kappa = 23.53$) is relatively inferior compared to its performance on the SGF dataset (average $\kappa = 35.88$), but it can still achieve a fair level agreement with human judgment. Concretely, IZSL can attain a fair level of agreement with human judgment on all three quality dimensions, with a slighter better results on "Suggestion" ($\kappa = 24.62$) than "Problem" ($\kappa = 23.27$) and "Positive Tone" ($\kappa = 22.71$). In addition, there is also a considerable disparity between the average κ of IZSL (= 23.5) and the average κ between the human annotators (= 88.0). The "cold-start" performance of the evaluation model still has considerable scope for improvement.

Based on our observations, we speculate that the "cold-start" performance variations across different datasets and quality dimensions can be primarily attributed to the varying degrees of syntactic and semantic diversity present in the data. The superior performance on the SGF dataset is mainly due to the fact that the system-generated feedback tends to utilize similar syntactic structures (e.g., "I would suggest / have liked to see"), while peer-review feedback exhibits more varied and sometimes ungrammatical phrases. The differences among quality dimensions are likely influenced by similar factors, such as the diverse nature of "Problem" statements in terms of syntactic structures and content.

In summary, the experimental results indicate that, even without using any labeled data, the IZSL model can achieve a moderate level of agreement with human judgment on the SGF dataset. The model can also attain a fair level of agreement on the PRF dataset,

Table 7.2: Performance evaluation of BERT (baseline) and IZSL on the *Peer-review Feedback (PRF)* dataset. The best results in each setting are marked in bold. CI = 95%.

	Suggestion		Problem		Pos. Tone		Average	
	F_1	κ	F_1	κ	F_1	κ	F_1	κ
Training with 0 labeled data samples (zero-shot)								
1 IZSL	**61.24**±2.4	**24.62**±4.2	**60.93**±2.3	**23.27**±4.5	**54.46**±1.8	**22.71**±2.5	**58.88**	**23.53**
Training with 50 labeled data samples								
2 BERT	65.08±12.24	32.46±22.4	63.43±5.7	31.53±7.1	60.04±3.4	21.64±6.8	62.85	28.54
3 IZSL	**91.49**±1.2	**84.32**±3.1	**84.83**±1.9	**69.79**±3.8	**80.75**±3.3	**61.51**±6.7	**85.69**	**71.87**
Training with 100 labeled data samples								
4 BERT	70.79±1.1	42.31±2.1	69.55±9.9	41.55±18.0	67.25±5.2	34.68±10.3	69.20	39.51
5 IZSL	**92.25**±1.5	**84.51**±3.0	**87.02**±2.8	**74.08**±5.5	**82.68**±2.1	**65.41**±4.1	**87.32**	**74.67**
Training with 250 labeled data samples								
6 BERT	77.87±7.4	56.3±14.14	83.13±7.1	66.35±14.0	79.61±5.5	59.29±10.8	80.21	60.65
7 IZSL	**93.49**±1.5	**86.99**±2.9	**87.90**±0.8	**75.86**±1.7	**85.07**±1.3	**70.19**±2.5	**88.82**	**77.68**
Training with 500 labeled data samples								
8 BERT	81.09±7.2	62.47±13.9	87.31±1.2	74.64±2.4	83.24±1.5	66.52±2.9	83.88	67.88
9 IZSL	**93.47**±0.7	**86.94**±1.5	**89.06**±1.0	**78.12**±2.0	**85.15**±1.4	**70.32**±2.7	**89.23**	**78.46**
Training with 750 labeled data samples								
10 BERT	90.69±0.5	81.39±1.1	87.80±4.3	75.58±8.5	83.45±3.2	66.93±6.3	87.32	74.63
11 IZSL	**93.71**±1.9	**87.42**±3.8	**90.22**±0.4	**80.46**±0.8	**85.51**±1.8	**71.04**±3.6	**89.81**	**79.64**
Training with 1000 labeled data samples								
12 BERT	91.70±0.9	83.42±1.9	88.81±0.6	77.63±1.3	84.85±1.5	69.73±3.0	88.46	76.93
13 IZSL	**93.77**±0.8	**87.54**±1.7	**90.37**±0.8	**80.76**±1.7	**86.37**±1.6	**72.74**±3.3	**90.17**	**80.35**

which exhibits greater syntactic and semantic diversity. Moreover, the IZSL's performance varies across different quality dimensions on both datasets. Despite these findings, there remains a significant gap between the "cold-start" performance of IZSL and the agreement between the human annotators. Further research is still required to achieve the ambitious objective of effectively evaluating feedback quality without using any labeled data.

7.3.2 Study 2: Performance in the incremental learning phase

The purpose of the second experiment is to ascertain whether the IZSL model can more efficiently evaluate feedback quality. In other words, we seek to investigate whether IZSL can surpass previous methods after training on incrementally collected data. The evaluation outcomes are shown in the second and subsequent rows of Table 7.1 and Table 7.2.

For the SGF dataset (Table 7.1), IZSL demonstrates exceptional performance. When trained with 50, 100, and 150 labeled samples, IZSL outperforms BERT with gaps of 18.19, 5.87, and 2.02 in the average κ, respectively. This indicates IZSL has stronger generalization ability, achieving near-saturating performance with minimal labeled samples. Moreover, with a mere 50 samples, the average κ of IZSL (= 85.98) reaches almost-perfect agreement ($81 < \kappa < 100$) with human judgment. With 150 samples, the average κ of IZSL (= 93.3) surpasses the average κ between the human annotators (= 92.0). This implies that IZSL can effectively mimic human evaluators in assessing the system-generated feedback.

For the PRF dataset (Table 7.2), we can obtain similar findings as for the SGF dataset. IZSL continues to demonstrate significantly superior generalization ability compared to BERT. For instance, with only 50 labeled data samples, the average κ of IZSL (= 71.87) can achieve substantial agreement ($61 < \kappa < 80$) with human judgment. In contrast, under the same setting, the average κ of BERT (= 28.54) can only reach a fair level of agreement ($21 < \kappa < 40$). Furthermore, with 1000 samples, the average κ of IZSL (= 80.35) nearly reaches an almost-perfect level of agreement ($81 < \kappa < 100$) with human judgment. Nevertheless, this average κ remains lower than the average κ between the human annotators (= 88.0).

We conjecture that one of the primary reasons for the stronger generalization ability of IZSL compared to BERT lies in our conversion of the feedback evaluation task into an NLI task. This adaptation facilitates a stronger alignment between our downstream task and the pre-training task of the employed BART checkpoint (Yin et al. 2019). The pre-training task of BART involves determining the logical relationship between two sentences, which bears a significant resemblance to our converted task. In contrast, the pre-training tasks of BERT are masked language modeling (MLM) and next sentence prediction (NSP) (Devlin et al. 2019), which exhibits weaker associations with the task of feedback evaluation.

In conclusion, the experimental results unequivocally demonstrate that IZSL exhibits considerably stronger generalization ability compared to the previous BERT-based models. In other words, IZSL can effectively mimic human evaluators in judging the feedback quality using significantly fewer labeled samples than those required by BERT. Remarkably, with only 150 samples from the SGF dataset, the average κ of IZSL surpasses the the average κ between the human annotators. Moreover, we hypothesize that the exceptional performance of IZSL lies in the alignment between the converted feedback evaluation task and the pre-training task. This hypothesis needs to be verified by extensive experiments.

7.4 Conclusion, Limitations, and Future Research

A major obstacle in the development of feedback-generation systems is the lack of efficient methods for evaluating the generated feedback. Chapter 6 has proposed using feedback-evaluation models to mimic human evaluators in judging the quality of feedback across different dimensions and demonstrated its feasibility. In this chapter, we introduce the incremental zero-shot learning (IZSL) model, which converts feedback evaluation into an NLI task, to reduce the model's reliance on annotations. The experimental results demonstrate that, on the two datasets, IZSL achieves acceptable performance in the "cold-start" phase and exhibits strong generalization ability in the incremental-learning phase.

This study has three key limitations that warrant consideration. Firstly, although the IZSL model demonstrates remarkable performance after fine-tuning with a small amount of human-judgment labels, there is still potential for enhancing its "cold-start" performance. Secondly, our approach to formulating hypotheses (i.e., prompts) relies predominantly on trial-and-error and experimentation on the validation set, which proves to be inefficient. Lastly, we hypothesize that IZSL's superior performance over BERT is primarily attributed to the closer alignment between the pre-training task of NLI-BART and our specific task. However, comprehensive experiments are necessary to substantiate this conjecture.

Looking ahead, we firmly believe that a pivotal pathway towards establishing feedback-evaluation models that harness zero labels involves integrating prompt engineering (Liu et al. 2023a) and large-scale language models (LLMs). Specifically, prompt engineering refers to the deliberate and strategic design of input queries or prompts given to out-of-the-box language models (e.g., ChatGPT) to elicit effective evaluation on specific dimensions of feedback quality. By crafting appropriate prompts to query these LLMs, we have the potential of achieving "zero-shot" evaluation of feedback. However, designing such prompts, especially ones that are applicable to different datasets and domains, is a challenge.

CHAPTER

8

On Assessing the Faithfulness of System-generated Feedback

8.1 Introduction

Feedback is integral in guiding students through their learning process, offering valuable insights that enable them to strengthen or correct their understanding of knowledge and content (Alharbi 2017; Hattie and Timperley 2007; Kusairi 2020; Race 2001). However, providing quality feedback on student assignments, particularly on open-ended and complex ones, presents a substantial challenge for educational resources and is often difficult to deliver in a timely manner. This demand for immediate and cost-efficient feedback has catalyzed interest in developing various automated feedback systems. Recent advancements in large language models (LLMs) have rendered LLM-based feedback systems the most prevalent medium for generating feedback (Jia et al. 2022a; Dai et al. 2023; Kasneci et al. 2023). In Chapter 4, we have demonstrated that the feedback produced by such LLM-based automated feedback systems is highly fluent, coherent, and plausible (Jia et al. 2022b).

However, it is also evident that LLM-based automated feedback systems are susceptible to generating spurious text (Ji et al. 2023). In other words, while these systems are capable of producing feedback that is often indistinguishable from instructor feedback in terms of fluency and coherence, they may generate prose that is erroneous, misleading, or entirely irrelevant to the input student assignments (Dai et al. 2023; Jia et al. 2022a,b). This phenomenon is commonly referred to as hallucination, which poses a substantial challenge that undermines the reliability of automated feedback systems (Maynez et al. 2020). Hallucinated content can potentially lead students to believe what is false, impacting their understanding of knowledge and even their motivation to learn (Hattie and Timperley 2007; Jia et al. 2022a). Therefore, understanding the hallucination problem and researching methods for its measurement are crucial to ensure the reliability of feedback systems.

In addition to the data-driven feedback systems introduced in Chapter 4, prompt-driven methods have emerged with the advent of ChatGPT (Team 2022). These two state-of-the-art approaches for implementing LLM-based feedback generation systems correspond to two distinct paradigms for customizing LLMs for feedback generation tasks: *pre-training and then fine-tuning* (Radford et al. 2018; Yang et al. 2019; Lewis et al. 2020b) and *prompt engineering* (Radford et al. 2019; Raffel et al. 2020; Gao et al. 2021; Dai et al. 2023). The former involves further training LLMs to capture underlying patterns from datasets of student work and instructor feedback to produce feedback (Jia et al. 2022b; Gombert et al. 2024). On the other hand, the latter entails leveraging human-crafted prompts to guide LLMs in generating desired feedback for student assignments (Dai et al. 2023; Phung et al. 2023). To gain a more comprehensive understanding of the hallucination problem in feedback generation, we use both systems to generate two sets of feedback on the same dataset.

In this chapter, our primary objective is to gain insight into the hallucination problem in feedback generation by thoroughly examining hallucinated content and exploring different methods for assessing the faithfulness of feedback. Specifically, we first implement a BART-based (Lewis et al. 2020b) data-driven feedback system (as introduced in Chapter 4) and an OpenAI ChatGPT4-based (Team 2022) prompt-driven feedback system, for automatically producing feedback on student project reports. Subsequently, we conduct a human evaluation of the feedback generated by the systems and analyze the types of hallucinated content in the feedback. Additionally, we investigate natural language inference (NLI)-based and OpenAI ChatGPT3.5-based approaches for measuring hallucinations in feedback. Lastly, we scrutinize whether hallucination raises ethical concerns in practical applications.

To investigate the hallucination problem in feedback generation, our research endeavors to answer the following research questions (RQs):

RQ1: Are both data-driven and prompt-driven automated feedback systems susceptible to hallucination? What is the prevalence of hallucinated content in the feedback?

RQ2: Is the hallucinated content produced by the systems primarily intrinsic (i.e., contrary to the input content) or extrinsic (i.e., neither supported nor contradicted)?

RQ3: Does the hallucinated content in feedback raise any ethical concerns? For example, does the hallucinated content contain offensive language or private information?

RQ4: Are there effective methods for measuring the hallucinations? Can the accuracy of the measurements be improved by utilizing human annotations on hallucination?

The main conclusions are as follows: First, both the data-driven and prompt-driven automated feedback systems generate a significant amount of hallucinated content (27.1% and 23.5%, respectively). Second, we observe that intrinsic hallucinations are more common in our data-driven system, whereas extrinsic hallucinations are more prevalent in our prompt-driven system. Third, when measuring hallucinations, instruction fine-tuned ChatGPT demonstrates the best alignment with human judgment (F_1 score $\approx$ 72%), yet still leaves considerable scope for improvement. Lastly, we fail to find any ethical concerns directly associated with hallucinations. The findings highlight opportunities to further research the mitigation and measurement of hallucinations in feedback generation.

Our main contributions are: 1) we analyze hallucinations in the feedback generated by two state-of-the-art LLM-based automated feedback systems for student project reports; 2) we collect a new dataset of hallucinations to facilitate future research endeavors; 3) we also implement NLI-based and ChatGPT3.5-based models for measuring hallucinations; 4) we highlight several prominent challenges for future studies in feedback generation.

The rest of this chapter is organized as follows: Section 8.2 first formulates the research problem, elaborates our methodology for investigating hallucination in feedback, and explains our methods for measuring hallucinations. Subsequently, Section 8.3 presents and discusses our experimental results. Finally, Section 8.4 concludes this chapter, acknowledges the limitations of our work, and provides some discussions on future research.

Table 8.1: Samples of instructor-provided and system-generated feedback for student project reports.

Feedback provided by the instructor: "The document is reasonably readable. The UI changes are described, but the code changes are not. The test plan is not adequately described. It should say what tests are to be performed, not just which functionality is to be tested. It is reasonable to test the UI manually, but model and controller tests should be automated."
Feedback from the data-driven system: "The writeup is very readable, but it does not explain the changes made to the code. It is not clear how the code is organized. The test plan is not described. This is a very important part of the design. I would have liked to see a description of how the tests work. Also, the test plan should have described how the tests work."
Feedback from the prompt-driven system: "The document outlines several improvements to the system's account request feature, including adding options for new institutions and introducing a user introduction text area. However, it lacks clarity in explaining how these changes are implemented and tested. The absence of detailed examples or code snippets makes it difficult to understand the modifications made to the system. Additionally, the document would benefit from a more organized structure, such as a table of contents, to improve readability and navigation."

8.2 Methodology

8.2.1 Problem Formulation

Automated feedback generation is often formulated as a *text-to-text generation* task, where the source text X represents input student work and the output target text is feedback Y. Then, feedback generation can be formally expressed as:

$$Y = \mathscr{F}_{\mathscr{M}}(X, \mathbb{C}) \tag{8.1}$$

where the model $\mathscr{F}_{\mathscr{M}}(\cdot)$ is usually powered by LLMs, and $\mathbb{C}$ is a group of desired properties (e.g., length). We consider that Y is hallucinated if it contradicts the information given in X, or the faithfulness of Y cannot be verified (i.e., neither supported nor refuted) by X.

Our objective is to gain insights into hallucination by examining the feedback generated by data-driven and prompt-driven systems. Furthermore, we aim to explore and develop methods for detecting hallucinated sentences (i.e., assessing the faithfulness) within Y.

8.2.2 Hallucinations in the Feedback

In addition to the data-driven system presented in Chapter 4, we implement a ChatGPT4-based prompt-driven system, similar to the one described in Dai et al. (2023). To guide the model in generating feedback for reports in the test set, we craft a few-shot prompt – *"You are an instructor responsible for providing feedback on student project reports. I will provide you with project reports and your task is to provide feedback for each of them. Here are five examples of project reports and paired feedback ... "* The five examples in the prompt are randomly selected from the training set. The feedback samples are shown in Table 8.1.

We then conduct human evaluation of feedback generated by both systems and analyze the types of hallucinated content. In the assessment, two human annotators were presented with student project reports and system-generated feedback. They were instructed only to assess the faithfulness of the feedback and not focus on other quality dimensions. The annotators are assigned to the responsibilities of 1) evaluating whether each feedback sentence contains hallucinated content, and 2) for every occurrence of hallucinated content, discerning whether the hallucination is intrinsic or extrinsic (defined in Section 2.4.1).

It is worth noting that hallucinations are not necessarily erroneous (Ji et al. 2023; Maynez et al. 2020). For certain extrinsic hallucinated content, although it is actually factual, it may not be faithful (i.e., it is hallucinated). For example, the feedback sentence *"This is a good example of a design doc, but it would have been better to show the code in a larger font."* is identified as an instance of extrinsic hallucination. While the font size of the code in the report is indeed small, there is no information regarding font size in the input to the feedback system. Thus, although it conforms to real-world facts (i.e., factual), this sentence remains unfaithful to the input content and thus it is labeled as extrinsic hallucination.

8.2.3 Measures for Hallucinations

Hallucination detection Dataset

To assess the performance of various methods in measuring hallucinations, we collect a new dataset consisting of feedback generated by the data-driven system. Specifically, by leveraging different combinations of decoding strategies and hyperparameters, we generate five collections of feedback for student project reports in the test set (n=50), thereby simulating a scenario that requires the measurement models for selecting hyperparameters.

The dataset consists of 250 feedback messages, with a total of 1430 sentences. The average word counts per feedback message and per sentence are 80 and 14, respectively. Approximately 29% of the sentences contain hallucinated content. We manually label the faithfulness of each sentence. The inter-annotator agreement between two annotators, measured by Cohen's κ coefficient, is 69.3%. This level of agreement is classified as substantial according to Cohen (1960), which demonstrates that our annotations are reliable.

NLI-based Method

We implement an NLI model using BART (Lewis et al. 2020b) and initialize all parameters with the "bart-large-mnli" checkpoint[1] (Yin et al. 2019), which is pretrained on the multi-genre NLI (MNLI) dataset. Using the pretrained weights provides us with an out-of-the-box NLI model for assessing faithfulness. In order to judge the faithfulness of generated feedback, we treat the source content (e.g., a student project report) as the premise, and each sentence from the feedback as the hypothesis. We first test it as an off-the-shelf classifier, and then we further assess it after supervised fine-tuning using human-annotated data.

ChatGPT3.5-based Method

We also leverage a ChatGPT3.5-based model for measuring hallucinations. Specifically, we utilize the "ChatGPT3.5-turbo-1106" model.[2] We experiment with various prompts and ultimately select the one (as shown in Appendix B) that demonstrates the best performance on the validation set in terms of the macro F_1 score under the one-shot condition (i.e., with a single example provided in the prompt). Similar to the use of the NLI-based model, we initially evaluate this model as an off-the-shelf classifier. Subsequently, we instruction fine-tune the model using human-annotated data and assess its performance again.

8.2.4 Experimental Setup

Training Details

The NLI model is trained on an NVIDIA A100 GPU (40GB) with a batch size of 4, a learning rate of 5e-5, epochs of 5 with early stopping, and the AdamW optimizer (Loshchilov and Hutter 2018) with a weight decay of 0.01. The ChatGPT3.5 model is trained using the OpenAI

[1] bart-large-mnli checkpoint: `https://huggingface.co/facebook/bart-large-mnli`
[2] ChatGPT3.5-turbo-1106: `https://platform.openai.com/docs/models/gpt-3-5-turbo`

ChatGPT API, with a temperature of 0 to reduce randomness and ensure deterministic outputs. The learning rate, batch size, and number of epochs are automatically set.

Evaluation metrics

We utilize the macro-F_1 score to evaluate models since we define the task of measuring hallucinations as a binary classification. The macro-F_1 score is the unweighted mean of F_1 scores calculated per class, so there is no distinction between highly and poorly populated classes. Thus, the macro-F_1 score provides a better picture of whether the model performs well on all classes than the micro-F_1 score (Grandini et al. 2020). We report the macro-F_1 score as well as its breakdowns on hallucination and non-hallucination since the dataset is skewed.

8.3 Experimental Results and Discussion

In this section, we present experimental results and provide answers to the research questions (RQs) stated earlier.

Table 8.2: Sample intrinsic and extrinsic hallucinations with explanations. Hallucinated content is indicated by wavy underlines.

Intrinsic hallucination example 1: "It is not clear what the changes are, and it would be helpful to have a more detailed description of them." (**Explanation:** This project report has included a list of changes, although it is not detailed.)
Intrinsic hallucination example 2: "Additionally, the document would benefit from a more organized structure, such as a table of contents, to improve readability and navigation." (**Explanation:** The project report has a table of contents.)
Extrinsic hallucination example 1 "Also, the screenshots are not very large, and the code snippets are not clearly separated." (**Explanation:** There is no image in the input doc, so the faithfulness of this text is neither supported nor refuted.)
Extrinsic hallucination example 2 "Finally, the document should consider providing examples or mock-ups of the proposed changes to facilitate better understanding and feedback from stakeholders." (**Explanation:** This is irrelevant text.)

Table 8.3: Percentages of hallucinated (intrinsic and extrinsic) content in generated feedback. Lower numbers for hallucinations and the higher number for faithfulness are boldfaced.

Method	Hallucination %		Faithfulness %
	Intrinsic	Extrinsic	
Data-driven	16.0	**11.1**	72.9
Prompt-driven	**9.8**	13.7	**76.5**

8.3.1 Study 1: Prevalence of hallucinated content

RQ1: Are both data-driven and prompt-driven automated feedback systems susceptible to hallucination? What is the prevalence of hallucinated content in the feedback?

The first experiment aims to determine whether both data-driven and prompt-driven automated feedback systems are prone to hallucination. Table 8.2 shows samples of hallucinations, and Table 8.3 presents the results of human evaluation. The prevalence of hallucinated content (intrinsic + extrinsic) in the feedback generated by data-driven and prompt-driven systems is 27.1% and 23.5%, respectively. That is, hallucinations occur in about a quarter of the sentences.

Overall, the results suggest that both systems are susceptible to generating hallucinated content. However, the prompt-driven system outperforms the data-driven system in terms of faithfulness, with a gap of 3.6%. It is also worth noting that the prompt-driven system tends to first summarize the project report, and then provide feedback. However, the data-driven system does not have this tendency.

RQ2: Is the hallucinated content produced by the systems primarily intrinsic (i.e., contrary to the input content) or extrinsic (i.e., neither supported nor contradicted)?

In the second experiment, we aim to explore the types (intrinsic or extrinsic) of hallucinated content in the feedback. As shown in Table 8.3, the percentages of intrinsic and extrinsic hallucinations in the feedback generated by the data-driven system are 16.0% and 11.1%, respectively. However, the feedback from the prompt-driven system contains 9.8% intrinsic hallucinations and 13.7% extrinsic hallucinations.

The results suggest that intrinsic hallucinations are more common in the data-driven system, while extrinsic hallucinations are more prevalent in the prompt-driven system.

We speculate that the data-driven system misinterprets spurious correlations in the data, leading to more feedback that contradicts the input information. The prompt-driven system, on the other hand, relatively lacks domain knowledge, making it more susceptible to generating irrelevant content.

RQ3: Does the hallucinated content in feedback raise any ethical concerns? For example, does the hallucinated content contain offensive language or private information?

The objective of the third experiment is to investigate whether the hallucinated content in feedback engenders any ethical concerns, particularly concerning the presence of offensive language or private information. As we mentioned, systematic methods are still lacking for assessing potential ethical issues arising from hallucinations. Consequently, we manually examined all system-generated feedback, and found no instances of the aforementioned ethical transgressions.

8.3.2 Study 2: Measuring the hallucinations

Table 8.4: Macro-F_1 scores and their breakdown on F_1-Hallucination and F_1-non-Hallucination for models measuring hallucinations. For each score, we report a 95% confidence interval (CI). The highest F_1 score in each column is boldfaced.

Method	**NonHal**	**Hal**	F_1
NLI	76.8±3.2	39.4±5.8	58.1±2.9
Fine-tuned NLI	85.4±3.5	54.0±6.3	69.8±3.2
ChatGPT3.5	70.8±3.7	48.8±4.9	59.9±3.6
Fine-tuned ChatGPT3.5	**86.9**±4.7	**57.2**±6.6	**72.1**±3.9

RQ4: Are there effective methods for measuring the hallucinations? Can the accuracy of the measurements be improved by utilizing human annotations on hallucination?

We now turn to the experimental results of measuring the hallucinations. As shown in Table 8.4, the out-of-the-box NLI and ChatGPT3.5 models achieve F_1 scores of 58.1% and 59.9%, respectively. However, the F_1 scores for the fine-tuned NLI and ChatGPT3.5 are 69.8% and 72.1%, which are improvements of 11.7% and 12.2%, respectively. Moreover, both models perform better on the non-hallucinated samples.

The results indicate that existing out-of-the-box LLMs still struggle to effectively judge the faithfulness of feedback. However, with the provision of additional human annotations, the performance of these models in measuring hallucinations can be significantly improved. Nonetheless, their performance is still not reliable enough for practical application, warranting further exploration and study. Therefore, measuring hallucinations remains a research opportunity.

8.4 Conclusion, Limitations, and Future Research

LLM-based automated feedback systems will play a crucial role in the future AI-powered educational ecosystem (Roschelle et al. 2020). However, concerns about hallucination impede the deployment of feedback systems in actual classroom settings. In this chapter, we have examined the feedback generated by both data-driven and prompt-driven systems for student project reports. The results demonstrate that both systems produce a considerable amount of intrinsic and extrinsic hallucinations. Moreover, the instruction fine-tuned ChatGPT achieves better alignment with human judgment in measuring hallucinations, yet more effective methods are still to be uncovered. This work contributes to a better understanding of hallucination in feedback generation, and our aspiration is to facilitate the practical deployment of feedback systems.

Limitations and future work: There are two main limitations to this study. First, while analyzing the types of hallucinated content in the feedback provided valuable insights, understanding the root causes of these hallucinations remains unexplored. Future research could deepen the understanding of hallucination issues by analyzing their sources. Second, we did not delve into how to effectively design prompts when using ChatGPT to measure hallucinations. Subsequent work could focus on creating more effective prompts to enhance the performance of hallucination detection. Addressing these limitations could contribute to refining the methodologies for evaluating and mitigating hallucinations in feedback systems, thereby paving the way for their more responsible application in education.

CHAPTER

9 Conclusion

The last chapter brings the dissertation to a close by presenting a comprehensive conclusion. In Section 9.1, we first summarize the key findings and insights garnered throughout our research endeavors. Section 9.2 then revisits the research questions (RQs) posed at the outset of this dissertation, explaining how our research has addressed and contributed to answering these questions. Subsequently, Section 9.3 thoroughly delineates the limitations of our research, acknowledging the constraints and potential implications for our findings. Lastly, Section 9.4 contemplates potential avenues for future research, outlining promising directions for further exploration and advancement in automated feedback systems.

9.1 Summary

This dissertation is dedicated to the automated provision of feedback for project reports. Part I (Chapter 4 and 5) focused on building data-driven feedback systems and understanding their practical implications and limitations. Part II (Chapter 6, 7, and 8) explored the efficient evaluation of feedback generated by the systems across various dimensions.

In Chapter 4, we introduced a data-driven automated feedback system, dubbed Insta-Reviewer. Our evaluation revealed that the feedback generated by this system is fluent, coherent, positive, and adept at identifying problems in student reports. However, there were instances of non-faithful or ambiguous statements in the generated feedback, and the system lacked proficiency in providing suggestions. Furthermore, no ethical issues were identified in the generated feedback. This work illustrates the feasibility of automated feedback generation for project reports and lays the groundwork for future endeavors.

In Chapter 5, we deployed Insta-Reviewer in a computer science course to understand its practical impact and limitations. We solicited student opinions on generated feedback through a questionnaire and conducted a semi-structured interview with the instructor. Their responses supported the effectiveness of the generated feedback in real classroom settings, but also raised concerns about faithfulness, project-specificity, and the need for the system to address follow-up questions. This study provides directions for future research to better align feedback systems with the pedagogical needs of students and instructors.

In Chapter 6, we presented a BERT-based multi-task learning (MTL) model to concurrently assess feedback quality across multiple dimensions (viz., containing suggestions, mentioning problems, and adopting positive tone). Experimental results suggested that jointly learning related tasks could improve the average performance, and our model achieved an almost-perfect level of agreement with human evaluators on all three dimensions. This study demonstrates the viability of leveraging limited data to build feedback-evaluation models that mimic human evaluators in assessing the quality of feedback.

In Chapter 7, we showcased a paradigm called incremental zero-shot learning (IZSL). This approach aimed to further reduce reliance on labels, even utilizing zero labels, for assessing feedback quality. The results indicated that, even without using any labeled data, IZSL achieved a fair level of agreement with human judgment. Moreover, compared to BERT-based models trained on the same data, IZSL could emulate human evaluators in assessing feedback quality using significantly fewer labeled samples. However, creating effective feedback-evaluation models that use zero labels remains an open challenge.

In Chapter 8, we investigated the hallucination problem in feedback generation (thereby assessing another crucial quality dimension—faithfulness). Our observation revealed that both the data-driven and prompt-driven automated feedback systems produce a considerable amount of hallucinated content. Furthermore, in assessing faithfulness, instruction fine-tuned ChatGPT exhibited the best alignment with human judgment, yet it remains unable to fully mimic human evaluators. This study advances the understanding of hallucination in feedback generation and investigates the methods for assessing faithfulness.

In summary, our efforts have demonstrated the feasibility of automated feedback generation for project reports. However, challenges such as ensuring faithfulness and offering project-specific suggestions still persist. Furthermore, our feedback-evaluation models have shown promising results in mimicking human evaluators to assess feedback quality. Nonetheless, the results also stress the challenges in constructing label-free evaluation models and measuring hallucinations. We encourage other researchers to address these difficulties to better support students and instructors in practical educational settings.

9.2 Research Questions Revisited

In this section, we revisit the research questions presented at the outset of this dissertation and describe how our research has addressed and contributed to answering the RQs:

RQ1: *Can we automatically generate feedback for student project reports? What are the potential deficiencies of the system-generated feedback?*

This research question was addressed in Chapter 4, where we thoroughly discussed the methodology and evaluation results of the feedback-generation system, Insta-Reviewer. Our research has contributed to addressing this question by implementing Insta-Reviewer and manually evaluating its effectiveness in generating feedback for student project reports. The system demonstrated the feasibility of automated feedback generation for student project reports. Notably, the system exhibited performance comparable to instructor feedback in terms of fluency, coherence, utilization of positive tone, and problem identification. However, the main deficiencies of the system-generated feedback involved its inclusion of hallucinated content and the limited ability to provide constructive suggestions to students. In addition, the generated feedback demonstrated no apparent ethical complications.

RQ2: *Do students appreciate system-generated feedback in actual classes? What may hinder the deployment of the automated feedback system?*

This research question was discussed in Chapter 5, where we evaluated Insta-Reviewer in a graduate-level computer science course to explore its implications and limitations. Our work has contributed to answering this question by deploying Insta-Reviewer in a real class and collecting authentic user experience. Student and instructor responses supported the effectiveness of the system-generated feedback in real educational settings. However, they also identified several barriers to practical deployment. First, students expressed concerns about the presence of unfaithful content in the feedback. Second, they noted that the generated feedback tended to be generic. Third, students perceived that generated feedback still cannot fully replace the nuanced insights provided by human instructors. Lastly, they expressed a desire for an interactive system capable of responding follow-up inquiries.

RQ3: *Can we build machine-learning models to mimic human evaluators in order to automate the evaluation of feedback?*

This research question was investigated in Chapters 6, 7, and 8, where we presented different models for mimicking human evaluators to assess feedback quality. Our studies made strides in addressing this question by introducing a BERT-based multi-task learning (MTL) model, a BART-based incremental zero-shot learning (IZSL) paradigm, and a ChatGPT-based faithfulness-evaluation model, to assess different quality dimensions. The results demonstrated that our models achieved an almost-perfect level of agreement with human evaluators on assessing “containing suggestions," “mentioning problems," and "adopting positive tone." However, for assessing the faithfulness of feedback, while the instruction-fine-tuned ChatGPT exhibited the best alignment with human evaluators, it still fell short of adequately replicating human judgment and necessitates future work.

9.3 Limitations

In this section, we delineate three primary limitations of this dissertation, acknowledging the constraints and potential implications for research findings and conclusions.

9.3.1 Data-driven Feedback Systems

The first limitation of our work was that we primarily focused on data-driven automated feedback systems. While this approach exhibits stronger generalization ability compared to rule-based methods, it heavily relies on historical data to fine-tune LLMs to generate feedback. However, the emergence of ChatGPT (Team 2022) has introduced a novel approach to producing feedback on student assignments, namely prompt-driven approach. This approach involves using expert-crafted prompts to guide LLMs to generate feedback. Although we have implemented such a system in Chapter 8 and analyzed the faithfulness of the feedback it generated, this dissertation did not systematically study this emerging approach. Nonetheless, the prompt-driven approach also has its own challenges, including the necessity for assignment-specific prompts and the absence of domain knowledge.

9.3.2 Self-reported In-class Evaluation

The second limitation of this dissertation was the reliance on anonymous questionnaire surveys and semi-structured interviews for collecting student and instructor opinions on the system-generated feedback. Although these methods provided valuable insights, the results could be influenced by biases or limitations inherent in self-reporting (Tourangeau et al. 2000). For example, there might be variations in how respondents interpret and respond to survey questions, resulting in response biases (Wetzel et al. 2016). Students might also be hesitant to criticize the system in the survey, as they are more likely to provide answers that they believe are socially acceptable, a phenomenon known as social desirability bias (Grimm 2010). In addition, questionnaire-based evaluation might not fully capture the nuanced perceptions of students, potentially resulting in incomplete conclusions.

9.3.3 Incomplete Evaluation Framework

The third limitation of our research was the potential incompleteness of the evaluation dimensions considered in assessing feedback quality. While our evaluation framework examined feedback across multiple dimensions, including suggestion provision, problem identification, utilization of positive tone, and faithfulness, it is important to acknowledge that these dimensions might not fully capture the nuances and complexities of feedback quality. For instance, our evaluation did not include an assessment of specificity, which refers to the degree to which feedback provides sufficiently specific and actionable guidance tailored to the individual project report. However, the interpretation of specificity can vary among different feedback recipients and evaluators, and the inherently subjective nature of specificity presents a formidable challenge in effectively automating its assessment.

9.4 Future Directions

Over the past four and a half years, we have been excited about the advancements in this field and delighted to have been actively involved. We strongly encourage more researchers to engage in the study of automated feedback-generation systems or to adopt such systems in classrooms. However, we also firmly believe that a complete replacement of manual feedback still requires a considerable amount of research and continues to face numerous challenges. At the end of this dissertation, we outline promising directions for further exploration. We believe this will contribute to the future endeavors of the community.

9.4.1 Prompt-driven Feedback Systems

With the development of general-purpose LLMs such as ChatGPT (Team 2022), prompt-driven methods have emerged. Instead of learning feedback-generation rules from paired data, these methods rely on expert-designed prompts, a practice known as prompt engineering (Liu et al. 2023a), to guide LLMs in leveraging their internal knowledge to generate feedback. For instance, Dai et al. (2023) investigated the viability of utilizing ChatGPT to generate feedback on student proposals. These prompt-driven systems offer potential advantages such as reducing reliance on annotated data, enhancing flexibility in tailoring the feedback, and providing personalized feedback to better accommodate the needs of diverse students. However, they also face challenges such as sensitivity to the wording of the prompt (Leidinger et al. 2023) and absence of domain knowledge (Liu et al. 2023b).

Two potential directions for improvement are *retrieval augmented generation (RAG)* (Lewis et al. 2020c) and *LLM agents* (Wang et al. 2024a). 1) RAG techniques integrate retrieval mechanisms with generation models, allowing them to access and incorporate external knowledge during the generation process. This can potentially mitigate the issue of domain-specific knowledge absence by enabling LLMs to access relevant information beyond their internal knowledge. 2) LLM agents involve using an architecture that combines LLMs with other modules like planning and memory. LLMs serve as the primary controller or "brain" that orchestrates a flow of operations necessary to complete tasks, such as feedback generation. By incorporating planning and memory modules, these LLM agents can enhance the capacity of systems to reason, plan, and adapt their feedback generation strategies.

9.4.2 Multi-modal Feedback Systems

Multi-modal feedback systems represent another promising direction for future research. Emerging feedback-generation systems are predominantly built upon LLMs that exclusively process textual information. This limitation results in feedback systems being constrained to using text modality alone as input to provide feedback, disregarding valuable information from other modalities, such as images and code (Jia et al. 2022a; Dai et al. 2023). By integrating information from different modalities, multi-modal feedback systems may offer more comprehensive and accurate feedback, thereby providing more helpful guidance for students (Wang et al. 2024b). However, the construction of multi-modal feedback systems also presents challenges, such as designing architectures and loss functions that effectively align and fuse information from different modalities (Wang et al. 2023; Li et al. 2024).

Two promising techniques for fusing information from different modalities are *early fusion* and *late fusion* (Gadzicki et al. 2020). 1) Early fusion involves concatenating the individual modalities into a unified representation at the input level, which is then processed as a single combined input (Chen et al. 2020; Li et al. 2020). This approach is relatively more parameter-efficient, but may be sensitive to data quality. 2) Late fusion treats each modality individually, and fuses information after separate processing in different uni-modal streams (Lu et al. 2019). While this approach allows for better capturing unique characteristics of each modality, it may have limited potential for exploiting correlations across modalities (Chen et al. 2023). By effectively fusing information from different modalities, multi-modal feedback systems have the potential to provide more valuable feedback to students.

9.4.3 Reference-free and Label-free Feedback Evaluation

Lastly, another important future direction lies in the development of reference-free and label-free feedback-evaluation methods. In this dissertation, we employed both automatic metrics and machine-learning models to assess feedback quality. However, the former assesses feedback quality by measuring the similarity between generated feedback and instructor feedback using ROUGE scores (Lin 2004) and BERTScore (Zhang et al. 2020), both of which require expensive expert-provided feedback as references. The latter requires a certain amount of human annotation to train the feedback-evaluation model to mimic human evaluators in assessing feedback quality. Such reliance on reference feedback or human annotations restricts the applicability of these feedback-evaluation methods.

A promising approach to establishing reference-free and label-free feedback evaluation involves integrating prompt engineering (Liu et al. 2023a) and out-of-the-box LLMs, such as ChatGPT (Team 2022). Appropriate human-designed prompts can effectively guide the LLMs to assess feedback quality across different dimensions without the need for reference feedback and human annotations. However, the performance of LLMs significantly depends on the quality of the prompt used to steer the model (Liu et al. 2023a). Researchers must experiment with various prompts to elicit desired behaviors (Kojima et al. 2022). To reduce the human effort in crafting prompts, Zhou et al. (2022) has proposed an *automatic prompt engineer* for automatically generating and selecting prompts. Future research may continue to explore this direction to achieve reference-free and label-free feedback evaluation.

REFERENCES

Ahea, M., Ahea, M., Kabir, R., Rahman, I., et al. (2016). The value and effectiveness of feedback in improving students' learning and professionalizing teaching in higher education. Journal of Education and Practice, 7(16):38–41.

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 Rafferty, A. N., Jacob Whitehill, C. R., and Cavalli-Sforza, V., editors, Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020), pages 565–568, Montreal, Canada. International Educational Data Mining Society.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

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, volume 1, pages 610–623, New York, NY. Association for Computing Machinery.

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, pages 133–145. Springer.

Branford, J., BROWN, A., and COCKING, R. (2000). How people learning: brain mind, experience and school. National Academy of Sciences. National Academy Press, Washington, DC.

Bulut, O. and Wongvorachan, T. (2022). Feedback generation through artificial intelligence. In The Open/Technology in Education, Society, and Scholarship Association Conference, volume 2, pages 1–9.

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 McCarthy, D. and Wintner, S., editors, 11th conference of the european chapter of the association for computational linguistics, pages 249–256, Trento, Italy. Association for Computational Linguistics.

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

Caruana, R. (1997). Multitask learning. Machine learning, 28(1):41–75.

Cavalcanti, A. P., Barbosa, A., Carvalho, R., Freitas, F., Tsai, Y.-S., Gašević, D., and Mello, R. F. (2021). Automatic feedback in online learning environments: A systematic literature review. Computers and Education: Artificial Intelligence, 2:100027.

Celikyilmaz, A., Clark, E., and Gao, J. (2020). Evaluation of text generation: A survey. arXiv preprint arXiv:2006.14799.

Chen, F.-L., Zhang, D.-Z., Han, M.-L., Chen, X.-Y., Shi, J., Xu, S., and Xu, B. (2023). Vlp: A survey on vision-language pre-training. Machine Intelligence Research, 20(1):38–56.

Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J. (2020). Uniter: Universal image-text representation learning. In European conference on computer vision, pages 104–120. Springer.

Chickering, A. W. and Gamson, Z. F. (1987). Seven principles for good practice in undergraduate education. AAHE bulletin, 3:7.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1):37–46.

Crawshaw, M. (2020). Multi-task learning with deep neural networks: A survey. arXiv preprint arXiv:2009.09796.

Dai, W., Lin, J., Jin, F., Li, T., Tsai, Y.-S., Gasevic, D., and Chen, G. (2023). Can large language models provide feedback to students? a case study on chatgpt. EdArXiv.

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 Burstein, J., Doran, C., and Solorio, T., editors, 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), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., Das, D., and Cohen, W. (2019). Handling divergent reference texts when evaluating table-to-text generation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4884–4895.

Durmus, E., He, H., and Diab, M. (2020). Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5055–5070.

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

Eddine, M. K., Shang, G., Tixier, A., and Vazirgiannis, M. (2022). Frugalscore: Learning cheaper, lighter and faster evaluation metrics for automatic text generation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1305–1318.

Elkan, C. (2001). The foundations of cost-sensitive learning. In International joint conference on artificial intelligence, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In Teh, Y. W. and Titterington, D. M., editors, Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 201–208, Sardinia, Italy. JMLR Workshop and Conference Proceedings.

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 Korhonen, A., Traum, D., and Màrquez, L., editors, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Feigenblat, G., Roitman, H., Boni, O., and Konopnicki, D. (2017). Unsupervised query-focused multi-document summarization using the cross entropy method. In Kando, N., Sakai, T., and Joho, H., editors, Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, page 961–964, New York, NY, USA. Association for Computing Machinery.

Filippova, K. (2020). Controlled hallucinations: Learning to generate faithfully from noisy data. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 864–870.

Gadzicki, K., Khamsehashari, R., and Zetzsche, C. (2020). Early vs late fusion in multimodal convolutional neural networks. In 2020 IEEE 23rd international conference on information fusion (FUSION), pages 1–6. IEEE.

Ganesan, K. (2006). Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. Computational Linguistics, 1(1).

Gao, L., Schulman, J., and Hilton, J. (2023). Scaling laws for reward model overoptimization. In International Conference on Machine Learning, pages 10835–10866. PMLR.

Gao, T., Fisch, A., and Chen, D. (2021). Making pre-trained language models better few-shot learners. In Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021, pages 3816–3830. Association for Computational Linguistics (ACL).

Garbacea, C. and Mei, Q. (2020). Neural language generation: Formulation, methods, and evaluation. arXiv preprint arXiv:2007.15780.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N. A. (2020). Realtoxicityprompts: Evaluating neural toxic degeneration in language models. arXiv preprint arXiv:2009.11462.

Gehringer, E. F. (2020). A course as ecosystem: melding teaching, research, and practice. In 2020 ASEE Virtual Annual Conference Content Access.

Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-up abstractive summarization. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Gombert, S., Fink, A., Giorgashvili, T., Jivet, I., Di Mitri, D., Yau, J., Frey, A., and Drachsler, H. (2024). From the automated assessment of student essay content to highly informative feedback: a case study. International Journal of Artificial Intelligence in Education, pages 1–39.

Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks.

Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.

Grimm, P. (2010). Social desirability bias. Wiley international encyclopedia of marketing.

Han, J., Yoo, H., Myung, J., Kim, M., Lim, H., Kim, Y., Lee, T. Y., Hong, H., Kim, J., Ahn, S.-Y., et al. (2023). Fabric: Automated scoring and feedback generation for essays. arXiv preprint arXiv:2310.05191.

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

Henry, E., Hinshaw, R., Al-Bataineh, A., and Bataineh, M. (2020). Exploring teacher and student perceptions on the use of digital conferencing tools when providing feedback in writing workshop. Turkish Online Journal of Educational Technology-TOJET, 19(3):41–50.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.

Honovich, O., Choshen, L., Aharoni, R., Neeman, E., Szpektor, I., and Abend, O. (2021). Q2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7856–7870.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

Huang, Y., Feng, X., Feng, X., and Qin, B. (2021). The factual inconsistency problem in abstractive text summarization: A survey. arXiv preprint arXiv:2104.14839.

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.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456. pmlr.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12):1–38.

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 Hsiao, I.-H. S., Sahebi, S. S., Bouchet, F., and Vie, J.-J., editors, Proceedings of the 14th International Conference on Educational Data Mining, pages 474–479. International Educational Data Mining Society.

Jia, Q., Young, M., Xiao, Y., Cui, J., Liu, C., Rashid, P., and Gehringer, E. (2022a). Insta-Reviewer: A Data-Driven Approach for Generating Instant Feedback on Students' Project Reports. In Mitrovic, A., Bosch, N., Cristea, A. I., and Brown, C., editors, Proceedings of the 15th International Conference on Educational Data Mining, number July in EDM '22, pages 5–16, Durham, United Kingdom. International Educational Data Mining Society.

Jia, Q., Young, M., Xiao, Y., Cui, J., Liu, C., Rashid, P., Gehringer, E., et al. (2022b). Automated feedback generation for student project reports: A data-driven approach. Journal of Educational Data Mining, 14(3):132–161.

Kallio, H., Pietilä, A.-M., Johnson, M., and Kangasniemi, M. (2016). Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. Journal of advanced nursing, 72(12):2954–2965.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. Learning and individual differences, 103:102274.

Keuning, H., Jeuring, J., and Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. ACM Transactions on Computing Education (TOCE), 19(1):1–43.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kitchenham, B. A. and Pfleeger, S. L. (2008). Personal opinion surveys. In Guide to advanced empirical software engineering, pages 63–92. Springer.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in neural information processing systems, 35:22199–22213.

Krishna, K., Roy, A., and Iyyer, M. (2021). Hurdles to progress in long-form question answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

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.

Lei, D., Li, Y., Wang, M., Yun, V., Ching, E., Kamal, E., et al. (2023). Chain of natural language inference for reducing large language model ungrounded hallucinations. arXiv preprint arXiv:2310.03951.

Leidinger, A., van Rooij, R., and Shutova, E. (2023). The language of prompting: What linguistic properties make a prompt successful? In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 9210–9232.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020a). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, page 7871–7880, Seattle, Washington. Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020b). 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, pages 7871–7880.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020c). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33:9459–9474.

Li, G., Duan, N., Fang, Y., Gong, M., and Jiang, D. (2020). Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 11336–11344.

Li, H., Li, C., Xing, W., Baral, S., and Heffernan, N. (2024). Automated feedback for student math responses based on multi-modality and fine-tuning. In Proceedings of the 14th Learning Analytics and Knowledge Conference, pages 763–770.

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

Li, S., Li, X., Shang, L., Dong, Z., Sun, C.-J., Liu, B., Ji, Z., Jiang, X., and Liu, Q. (2022b). How pre-trained language models capture factual knowledge? a causal-inspired analysis. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1720–1732.

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

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023a). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9):1–35.

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., et al. (2023b). Summary of chatgpt-related research and perspective towards the future of large language models. Meta-Radiology, page 100017.

Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In International Conference on Learning Representations.

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, Vancouver, BC, Canada. OpenReview.net.

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

Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. Advances in neural information processing systems, 32.

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

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, pages 790–795, online. International Educational Data Mining Society.

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, pages 1–7, Seattle, United States. Association for Computational Linguistics.

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. International Educational Data Mining Society.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1906–1919.

McNichols, H., Feng, W., Lee, J., Scarlatos, A., Smith, D., Woodhead, S., and Lan, A. (2023). Exploring automated distractor and feedback generation for math multiple-choice questions via in-context learning. arXiv preprint arXiv:2308.03234.

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.

Mündler, N., He, J., Jenko, S., and Vechev, M. (2023). Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. arXiv preprint arXiv:2305.15852.

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

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

Navigli, R., Conia, S., and Ross, B. (2023). Biases in large language models: Origins, inventory and discussion. ACM Journal of Data and Information Quality.

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

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

Pahari, N. and Shimada, K. (2022). Multi-task learning using bert with soft parameter sharing between layers. In 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS), pages 1–6. IEEE.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In Isabelle, P., editor, Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA. The association for computational linguistics.

Parikh, A., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. (2020a). Totto: A controlled table-to-text generation dataset. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1173–1186.

Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., Das, D., and Tech, G. (2020b). Totto: A controlled table-to-text generation dataset.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543.

Phung, T., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., Singla, A., and Soares, G. (2023). Generating high-precision feedback for programming syntax errors using large language models. arXiv preprint arXiv:2302.04662.

Pilault, J., Li, R., Subramanian, S., and Pal, C. (2020). On extractive and abstractive neural document summarization with transformer language models. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9308–9319, Online. Association for Computational Linguistics.

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

Prechelt, L. (2002). Early stopping-but when? In Neural Networks: Tricks of the trade, pages 55–69. Springer.

Race, P. (2001). Using feedback to help students to learn. The Higher Education Academy.

Race, P. (2019). The lecturer's toolkit: a practical guide to assessment, learning and teaching. Routledge.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

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.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In 4th International Conference on Learning Representations, ICLR 2016.

Reiter, E. and Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. Computational Linguistics, 35(4):529–558.

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.

Rumelhart, D. E. (1986). Learning internal representations by error propagation, in parallel distributed processing. Explorations in the Microstructure of Cognition, pages 318–362.

Scherer, T., Straub, J., Schnyder, D., and Schaffner, N. (2013). The effects of anonymity on student ratings of teaching and course quality in a bachelor degree programme. GMS Zeitschrift für Medizinische Ausbildung, 30(3).

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, pages 7881–7892, Online. Association for Computational Linguistics.

Shute, V. J. (2008). Focus on formative feedback. Review of educational research, 78(1):153–189.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958.

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification? arXiv preprint arXiv:1905.05583.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

Tamkin, A., Handa, K., Shrestha, A., and Goodman, N. (2022). Task ambiguity in humans and language models. arXiv preprint arXiv:2212.10711.

Tang, G., Müller, M., Gonzales, A. R., and Sennrich, R. (2018). Why self-attention? a targeted evaluation of neural machine translation architectures. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4263–4272, Brussels, Belgium. Association for Computational Linguistics.

Team, O. (2022). Chatgpt: Optimizing language models for dialogue.

Tourangeau, R., Rips, L. J., and Rasinski, K. (2000). The psychology of survey response.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Tyagi, P. K. (1989). The effects of appeals, anonymity, and feedback on mail survey response patterns from salespeople. Journal of the Academy of Marketing Science, 17:235–241.

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

van der Lee, C., Gatt, A., van Miltenburg, E., and Krahmer, E. (2021). Human evaluation of automatically generated text: Current trends and best practice guidelines. Computer Speech & Language, 67:101151.

Van Der Lee, C., Gatt, A., Van Miltenburg, E., Wubben, S., and Krahmer, E. (2019). Best practices for the human evaluation of automatically generated text. In Proceedings of the 12th International Conference on Natural Language Generation, pages 355–368.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762.

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, Palais des Congrès Neptune, Toulon, France.

Wang, A., Cho, K., and Lewis, M. (2020). Asking and answering questions to evaluate the factual consistency of summaries. In 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, pages 5008–5020. Association for Computational Linguistics (ACL).

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. (2024a). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6):1–26.

Wang, S., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P. S., and Wen, Q. (2024b). Large language models for education: A survey and outlook. arXiv preprint arXiv:2403.18105.

Wang, W., Fraser, G., Barnes, T., Martens, C., and Price, T. (2021). Automated classification of visual, interactive programs using execution traces. In Hsiao, I.-H. S., Sahebi, S. S., Bouchet, F., and Vie, J.-J., editors, Proceedings of the 14th International Conference on Educational Data Mining, pages 677–681. International Educational Data Mining Society.

Wang, X., Chen, G., Qian, G., Gao, P., Wei, X.-Y., Wang, Y., Tian, Y., and Gao, W. (2023). Large-scale multi-modal pre-trained models: A comprehensive survey. Machine Intelligence Research, 20(4):447–482.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560.

Wetzel, E., Böhnke, J. R., and Brown, A. (2016). Response biases.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122. Association for Computational Linguistics.

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.

Winstone, N. E., Nash, R. A., Parker, M., and Rowntree, J. (2017). Supporting learners' agentic engagement with feedback: A systematic review and a taxonomy of recipience processes. Educational psychologist, 52(1):17–37.

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, pages 2071–2080.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

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 Rafferty, A. N., Jacob Whitehill, C. R., and Cavalli-Sforza, V., editors, Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020), pages 704–709, Montreal, Canada. International Educational Data Mining Society.

Xu, Z., Jain, S., and Kankanhalli, M. (2024). Hallucination is inevitable: An innate limitation of large language models. arXiv preprint arXiv:2401.11817.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.

Yin, W., Hay, J., and Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. arXiv preprint arXiv:1909.00161.

Yoon, S.-Y., Miszoglad, E., and Pierce, L. R. (2023). Evaluation of chatgpt feedback on ell writers' coherence and cohesion. arXiv preprint arXiv:2310.06505.

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.

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 Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada. Curran Associates Inc.

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.

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., et al. (2023). Siren's song in the ai ocean: A survey on hallucination in large language models. arXiv preprint arXiv:2309.01219.

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

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2022). Large language models are human-level prompt engineers. In The Eleventh International Conference on Learning Representations.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In The IEEE International Conference on Computer Vision (ICCV).

Zingle, G., Radhakrishnan, B., Xiao, Y., Gehringer, E., Xiao, Z., Pramudianto, F., Khurana, G., and Arnav, A. (2019). Detecting suggestions in peer assessments. In Lynch, C. F., Merceron, A., Desmarais, M., and Nkambou, R., editors, Proceedings of the 12th International Conference on Educational Data Mining, pages 474–479. International Educational Data Mining Society.

APPENDICES

APPENDIX

A

ACRONYMS

A summary of all acronyms is documented in Table A.1.

Table A.1: A summary of acronyms used in alphabetical order.

Acronym	Abbreviation
Bidirectional Encoder Representations from Transformers	BERT
Confidence Interval	CI
Database	DB
Diverse Beam Search	DBS
Gated Recurrent Unit	GRU
Global Vectors for Word Representation	GloVe
Incremental zero-shot learning	IZSL
Institutional Review Board	IRB
Long Short-Term Memory	LSTM
Masked Language Modeling	MLM
Multi-task Learning	MTL

Natural Language Generation	NLG
Natural Language Processing	NLP
Natural Language Inference	NLI
Next Sentence Prediction	NSP
Peer-review Feedback	PRF
Pre-trained Language Model	PLM
Question-answering	QA
Recurrent Neural Network	RNN
Research Questions	RQs
Retrieval Augmented Generation	RAG
Sequence-to-sequence	Seq2seq
Single-task Learning	STL
Student Project Reports and Feedback	SPRF
System-generated Feedback	SGF

APPENDIX

B

PROMPT USED IN THE WORK

We followed the prompt-engineering guidance[1] published by OpenAI ChatGPT to design the prompt as below.

system_in = "You will be provided with a student project report (delimited by XML tags <doc> and </doc>) and the corresponding textual feedback on that project report (delimited by XML tags <feedback> and </feedback>, with sentences separated using </sen>).

However, the provided feedback is generated by large language models. And it may not be entirely faithful (i.e., some sentences in the generated feedback may be hallucinated - irrelevant, made-up, or inconsistent with the project report).

Your task is to evaluate whether each sentence of the feedback is faithful (i.e., not hallucinated) to the corresponding student project reports. To accomplish this, you should read and understand the content of the reports, and then read each sentence of the feedback to evaluate whether the sentence is faithful.

Use labels to mark, 1 if the sentence is likely to be faithful, and 0 if the sentence is likely

[1]https://platform.openai.com/docs/guides/prompt-engineering

to be unfaithful. Please judge each sentence and return the original sentence and its label.

<An example is inserted here.>

For example, the feedback contains 6 sentences: 1. This is a very good description of the changes made to the code. 2. The design doc is very readable. 3. It explains the changes well. 4. However, it does not explain how the code was refactored, or how it was tested. 5. Also, the code is not consistent with the design doc. 6. This is a good thing.

If you think the sentences 1, 2, 3, 5 are faithful to the project report. And the sentences 4, 6 are unfaithful to the project report. You should return the original sentences and labels like this:

1. This is a very good description of the changes made to the code. - 1
2. The design doc is very readable. - 1
3. It explains the changes well. - 1
4. However, it does not explain how the code was refactored, or how it was tested. - 0
5. Also, the code is not consistent with the design doc. - 1
6. This is a good thing. - 0

Please judge the sentences and follow the format above to return your answers."