

How Students Engage with LLM Feedback Matters: Feedback Processing Time and Learning in Introductory Programming

Maciej Pankiewicz¹[0000-0002-6945-0523], Conrad Borchers²[0000-0003-3437-8979] and Ryan S. Baker³[0000-0002-3051-3232]

¹ University of Pennsylvania, Philadelphia PA, USA
mpank@upenn.edu

² Vanderbilt University, Nashville, TN, USA

³ Adelaide University, Adelaide, Australia

Abstract. Large Language Models (LLMs) are increasingly being used to generate dynamic feedback in programming education, but limited research has examined how students’ engagement with LLM-generated feedback relates to learning outcomes. We test whether students’ feedback reading time, used as a proxy for feedback engagement, predicts learning outcomes, and whether this relationship differs when LLM feedback is available. Using log data from a randomized experiment in an introductory programming course, we trained and validated a ridge regression model to predict log-transformed feedback reading time from behavioral features in the treatment group and applied it to both conditions to estimate engagement propensity. Predicted feedback reading time positively predicted final test performance after controlling for prior knowledge, with no condition-by-engagement interaction. Exploratory analyses suggested possible diminishing returns at higher levels of reading time. These results suggest that learning outcomes depend less on the mere availability of LLM-generated feedback and more on how students engage with feedback, motivating adaptive designs that support deeper feedback processing.

Keywords: Automated feedback, LLM, Programming education.

1 Introduction

A useful way to frame LLM-based feedback in programming is through the long-standing literature on scaffolding and help use [1]. Intelligent tutoring systems have repeatedly demonstrated that step-level guidance can produce positive learning outcomes, but that learning gains depend on how learners engage with assistance. Research on cognitive tutors has provided important evidence for understanding how learning environments should balance the provision and withholding of information or assistance to maximize student learning. This challenge, known as the assistance dilemma [3], was originally studied in domains where learners’ actions could be interpreted against relatively well-defined solution paths.

In computer science (CS) education, however, this issue has been harder to examine using traditional tutor architectures. Although programming tutors [2] demonstrated that fine-grained tutoring is possible for constrained tasks, extending such approaches to larger, free-form introductory code-writing tasks is considerably more difficult.

Recent advances in large language models (LLMs), have created new possibilities for providing useful support in this context [4]. Although LLMs are commonly used through chat interfaces, they can also be incorporated into automated programming assessment systems as embedded feedback features [5]. In such environments, students develop code within an ongoing feedback loop, while scaffolding is provided through LLM-generated feedback.

In this study, students were assigned to an LLM-feedback condition or a control condition without LLM support in an automated programming assessment platform. Because feedback reading time is observable only for students who receive LLM feedback, we estimated each student's propensity to engage with feedback from other platform interaction features. Predicting this propensity for both conditions allowed us to test whether it predicts learning outcomes beyond prior knowledge, and whether this relationship differs when LLM feedback is available.

2 Methods

Data for this study was collected during the winter semester of 2024/2025 at a large European university. The participants were computer science students enrolled in a mandatory first-semester course titled "Introduction to Programming." The course focused on programming in C#. A total of 227 students consented to participate and completed a (non-graded) pre-test (eight items, one point each) and the final (graded) test at the end of the semester (nine items, one point each). Students used a platform for automated assessment of programming assignments throughout the semester, completing programming tasks on topics such as data types, variables, loops, and recursion. Usage of the platform was not mandatory, and performance did not count towards the course grade.

The platform's assessment process involved compiling and testing the submitted code. Regular platform feedback included compiler messages for non-compiling (containing syntax errors) code submissions. For submissions that were successfully compiled, students received results for each unit test executed on the code (with input values and expected outcome for each unit test), and overall score (the percentage of unit tests that ended successfully: 0-100%). Students were randomly assigned to either Control or Experimental (LLM) condition. Students in the Experimental condition additionally received feedback generated automatically using OpenAI's GPT-4o model for submissions containing compiler errors. Feedback was created via an API request using the approach to prompt generation that consisted of the following parts: 1) general instructions for an assistant, 2) text of a programming assignment, 3) student code, 4) list of detected compiler errors [5]. Feedback was displayed to students automatically as a pop-up immediately after submission. Once the pop-up was closed, the feedback remained accessible below the code editor for further review.

Because reaction and processing times can indicate engagement and performance in complex problem solving [7], we engineered behavioral features to predict students' feedback reading time. These included total submissions, non-compiling submissions, solved and attempted tasks, accuracy and success rates, total platform time, and average time per submission. Together, these features captured students' productivity, problem-solving efficiency, and platform engagement.

To estimate the propensity for students in the Control condition to have engaged with feedback had it been available, we conducted a predictive modeling analysis [8]. We trained a regularized linear model (ridge regression) to predict the log-transformed average feedback reading time within the Experimental group, as a proxy for engagement with the LLM-generated feedback. The primary justification for choosing ridge regression is its robustness and strong performance in settings with a limited number of data points (93 students in the Experimental group) relative to features (7 features), which is the case here. The ridge penalty mitigates potential multicollinearity among the behavioral features and ensures stable coefficient estimates. This model was validated using 5-fold cross-validation. We then applied the fitted model to the identical feature set from students in the Control group. This generated predicted feedback reading time for each control student.

The model was validated using a two-step approach. Initial 5-fold cross-validation on the Experimental group yielded a mean RMSE of 0.50 (SD=0.06) on the log-transformed target. To obtain robust uncertainty estimates, we performed bootstrap validation with 10,000 resamples. This confirmed the model's performance: the ridge regression achieved a mean RMSE of 0.44 (95% CI [0.36, 0.51]). By comparison, a simple mean baseline predictor, which ignores all features and always predicts the average reading time from the training data, had a mean RMSE of 0.60 (95% CI [0.52, 0.68]). A subsequent randomization test confirmed this difference was statistically significant ($p=.001$). This validated model was then applied to the Control group's feature set to generate predicted feedback reading time for each control student.

These predicted values allow for a comparative analysis of feedback engagement propensity between the Experimental and Control conditions. To investigate the relationship between predicted feedback reading time, prior knowledge, Experimental condition, and learning outcomes, we estimated a sequence of ordinary least squares regression models. The analysis proceeded in three steps. First, we estimated a baseline model regressing final test score on the predicted feedback reading time alone (Model 1). Second, we added pre-test score as a covariate to control for baseline programming knowledge (Model 2). Third, we tested two potential moderators: whether the effect of predicted engagement time on learning differed by condition (Model 3: propensity $\times$ Group + pre_test), and whether it interacted with prior knowledge (Model 3b: propensity $\times$ pre_test). These models allowed us to test whether the predictive value of estimated feedback processing depended on receiving the actual intervention or on a student's incoming skill level. The analysis employed four ordinary least squares regression models:

Model 1: Final Test Score = $\beta_0 + \beta_1(\text{Predicted Reading Time})$

Model 2: Final Test Score = $\beta_0 + \beta_1(\text{Predicted Reading Time}) + \beta_2(\text{Pre-test Score})$

Model 3a: Final Test Score = $\beta_0 + \beta_1(\text{Predicted Reading Time}) + \beta_2(\text{Experimental Group}) + \beta_3(\text{Predicted Reading Time} \times \text{Experimental Group}) + \beta_4(\text{Pre-test Score})$

Model 3b: Final Test Score = $\beta_0 + \beta_1(\text{Predicted Reading Time}) + \beta_2(\text{Pre-test Score}) + \beta_3(\text{Predicted Reading Time} \times \text{Pre-test Score})$

In these equations, Experimental Group is a binary indicator (0=Control, 1=LLM Feedback). The interaction terms test whether the relationship between predicted reading time and test score differs by experimental condition (Model 3a) or depends on prior knowledge (Model 3b).

3 Results

We collected 62,221 code submissions to programming problems. Mann–Whitney U tests indicated no statistically significant differences between the Control and Experimental groups in pre-test or final test performance, nor in platform engagement measures, including submissions, solved problems, and the proportion of non-compiling submissions.

Feedback reading time in the Experimental condition was calculated as the difference between the timestamp when the feedback was displayed and when the student closed the pop-up. Student-level descriptive statistics showed that, on average, students received 94.6 feedback messages (SD=80.9) and spent 25.4 seconds (SD=20.03) reading. Within the Experimental condition, feedback reading time was positively associated with final test performance. In an OLS regression controlling for pre-test score, feedback reading time (in seconds) significantly predicted final test score ($\beta=0.038$, $p=.027$), as did pre-test score ($\beta=0.458$, $p<.001$), suggesting that students who spent more time reading LLM-generated feedback tended to perform better on the final test.

Model 1 showed no significant association between predicted log-transformed feedback reading time and final test score ($\beta=0.114$, $p=.675$). However, after controlling for prior knowledge in **Model 2**, predicted reading time became a significant positive predictor ($\beta=0.793$, $p=.003$), with pre-test score also a strong predictor ($\beta=0.450$, $p<.001$).

The moderation analyses yielded null results for the interaction hypotheses. In **Model 3a**, the interaction between predicted reading time and experimental group was non-significant ($\beta=-0.027$, $p=.956$), indicating the relationship between estimated reading time and performance did not differ between the Treatment and Control groups. The main effect of predicted reading time remained significant ($\beta=0.842$, $p=.022$). The direct effect of the Group variable was also non-significant ($\beta=-0.501$, $p=.742$).

In **Model 3b**, the interaction between predicted reading time and pre-test score was non-significant ($\beta=-0.003$, $p=.976$). In summary, predicted feedback reading time positively predicted final performance after accounting for prior knowledge (Fig. 1).

We compared models using both Akaike (AIC) and Bayesian (BIC) information criteria. **Model 2** (with pre-test as a covariate) showed substantial improvement over **Model 1**, with AIC decreasing from 868.4 to 825.6 and BIC from 874.9 to 835.5. According to BIC (which imposes a stronger penalty for model complexity) **Model 2** remained the most parsimonious (BIC=835.5), outperforming the more complex interaction models (BIC=841.6 for both Models 3a and 3b). Therefore, we conclude that

Model 2 describes our data best. Importantly, propensity did not differ between Experimental and Control groups (Welch’s $t=0.72$, $p=.47$, Cohen’s $d=0.10$), indicating that observed/predicted engagement was not confounded with experimental condition.

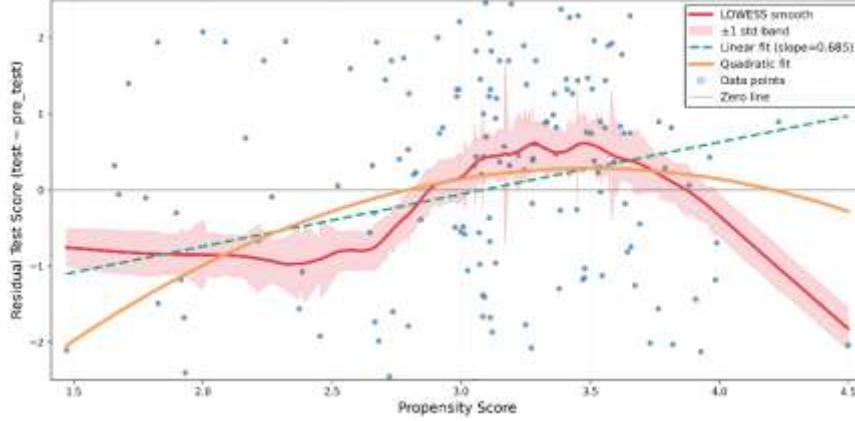


Fig. 1. Empirical relationship between predicted feedback reading time and test score

To further probe whether longer engagement times might reflect confusion or disengagement, we residualized post-test scores with respect to pre-test scores, yielding a measure of learning beyond prior knowledge. Spearman rank correlations showed no monotonic relationship between propensity and raw post-test performance ($\rho=-.01$, $p=.91$) but showed a positive association between propensity and residualized learning ($\rho=.17$, $p=.019$), suggesting that higher engagement propensity was associated with greater learning gains after accounting for prior knowledge. To assess potential non-linearities, we fit a quadratic+linear model predicting residualized learning (controlling for pre-test) from propensity. The linear term was marginally significant and positive ($\beta=3.97$, $p=.052$), while the quadratic term was negative and also marginally statistically significant ($\beta=-0.57$, $p=.098$). Still, as Fig. 1 shows, the marginally significant quadratic effect was mild in magnitude. The LOWESS curve indicates weak and asymmetric curvature, with lower performance observed at both ends of the engagement propensity distribution.

4 Discussion and Conclusion

This paper advances research on scaffolding and help use by showing that the impact of LLM-generated feedback depends on learner engagement. We introduce a scalable approach for inferring engagement with assistance from behavioral traces and connect LLM-based feedback systems to established principles of constructive help use. In our study, LLM feedback for compiler errors was neither uniformly beneficial nor harmful. Instead, students who spent more time reading feedback, and those with higher predicted engagement propensity, tended to perform better on the final test after controlling for prior knowledge. This pattern aligns with research on intelligent tutoring systems showing that assistance supports learning when processed constructively but may

be neutral or counterproductive when used instrumentally (e.g., *gaming the system* [6]). Here, reading time is a scalable but imperfect proxy for feedback processing: brief interactions may reflect superficial use, while longer engagement may indicate interpretation and repair, though very long times may also signal confusion or off-task behavior. Overall, the absence of a main treatment effect, combined with a positive engagement–outcome association, suggests conditional benefits: LLM feedback may help students who engage meaningfully with it, but its availability alone does not guarantee learning gains.

Several limitations warrant further study. Feedback reading time is an imperfect proxy for engagement because pop-up open time may include actual reading, pauses, multitasking, confusion, or leaving the feedback window open. Although our propensity model was validated against strong baselines, unobserved confounds such as motivation may remain. Future work should triangulate reading time with richer process measures, such as scrolling and post-feedback code edits, and experimentally test whether designs that promote constructive feedback use improve learning.

Overall, our findings suggest that the educational value of LLM-based programming feedback is conditional on how learners engage with it, and that scalable proxies for feedback use can help educational systems personalize assistance to increase benefits while reducing shallow or instrumental use of LLMs.

References

1. Aleven, V., Roll, I., McLaren, B.M., Koedinger, K.R.: Help helps, but only so much: Research on help seeking with intelligent tutoring systems. *Int. J. Artif. Intell. Educ.* 26, 205–223 (2016).
2. Anderson, J.R., Reiser, B.J.: The LISP tutor. *Byte*. 10, 4, 159–175 (1985).
3. Koedinger, K.R., Aleven, V.: Exploring the Assistance Dilemma in Experiments with Cognitive Tutors. *Educ. Psychol. Rev.* 19, 3, 239–264 (2007). doi:10.1007/s10648-007-9049-0.
4. Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., Becker, B.A.: Using Large Language Models to Enhance Programming Error Messages. In: *Proceedings of SIGCSE 2023*, pp. 563–569 (2023). doi:10.1145/3545945.3569770.
5. Pankiewicz, M. and Baker, R.S.: Enhancing Student Focus and Problem-Solving with Real-Time LLM Feedback on Compiler Errors. In: *Proceedings of EC-TEL 2025*. pp. 412–426 (2026). doi:10.1007/978-3-032-03870-8_28
6. Paquette, L., Baker, R.S.: Variations of Gaming Behaviors Across Populations of Students and Across Learning Environments. In: *Proceedings of AIED 2017*. pp. 274–286 (2017).
7. Scherer, R., Greiff, S., Hautamäki, J.: Exploring the Relation between Time on Task and Ability in Complex Problem Solving. *Intelligence*. 48, 37–50 (2015). doi:10.1016/j.intell.2014.10.003.
8. Thomas, D. R., Borchers, C., Bhushan, S., Gatz, E., Gupta, S., Koedinger, K.R.: LLM-Generated Feedback Supports Learning if Learners Choose to Use it. In: *Proceedings of EC-TEL 2025*, pp. 489–503 (2026). doi:10.1007/978-3-032-03870-8_33