Behavior Detectors to Support Feedback Generation using Problem-Solving Action Data

by

Aaron Alphonsus

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

May 2021

APPROVED:

Professor Neil T. Heffernan, Major Thesis Advisor

Professor Jacob R. Whitehill, Thesis Reader

Professor Craig E. Wills, Head of Department

Abstract

Feedback is an essential component of learning and a key difficulty in achieving quality education at scale. Providing feedback is often a tedious task and there is a paucity of resources to aid teachers. In this work, we expand on previous tools that focus on generating natural language feedback for open response questions. Computer-based systems have the unique advantage of being able to collect action-by-action reports of the steps a student took to reach an answer along with metadata, such as time spent on a problem. It is difficult for teachers to analyze the detailed metadata when providing feedback, but it presents us with an opportunity to distill information from it. We take on problem-solving action data to provide teachers with detectors of student behavior. These detectors can be used to better keep track of their students' activity and inform what feedback can be provided.

Acknowledgements

I want to express my gratitude to my advisor, Prof. Neil Heffernan for his guidance throughout my time at WPI. I want to thank my thesis reader, Prof. Jacob Whitehill for his feedback. I also want to thank Prof. Anthony Botelho for his continuous support and mentorship in this thesis work and my work at WPI.

A big thank you to the ASSISTments team at WPI and my peers in the lab for their feedback and guidance throughout. I also thank my family and friends who have been a constant source of support and strength.

And finally, to you. Wow, you're actually reading this. I am flattered and thankful for the support.

Contents

1 Introduction **1**

1.1 Background and Related Work 3

1.1.1 QUICK-Comments 3

1.1.2 LIVE-CHART 4

2 Evaluating NLP-based Feedback Models **6**

2.1 Evaluating Comment Suggestions 6

2.2 RCT Infrastructure for QUICK-Comments 8

3 Activity Detectors for Problem-Solving Action Data **10**

3.1 Dataset and pre-processing 11

3.1.1 Feature Creation 12

3.1.2 Maintaining Student Sequences Within Fold 12

3.1.3 Collapsing Categories 13

3.2 First Model: When to give Feedback? 13

3.3 Second Model: Detecting Positive Feedback 14

3.4 Results 15

4 Discussion **16**

A ERD for QUICK-Comments RCTs **18**

Chapter 1

Introduction

In numerous scenarios, but especially in education, feedback plays an important role in iterative improvement [2]. However, providing ample, good quality feedback remains one of the major pain points in achieving education at scale. Student evaluations routinely confirm this when they reveal their dissatisfaction with the amount of quality feedback in end of course surveys [1]. At scale, feedback is a difficult problem for teachers as it requires large amounts of manual work. Computer-based learning systems have been able to alleviate some of this by providing simple correctness feedback. While this is certainly important and shown to be beneficial [3], the real hard challenge is to provide automated grading and feedback that is close to human-level accuracy.

Learning systems have generally supported feedback for close-ended problem types with structured answers. These take the form of fill-in-the-blank or multiple choice questions. It is quite a bit harder to offer feedback support for open-ended, descriptive questions. A sample of the many challenges to providing feedback on open response problems are: (1) student responses are diverse, (2) responses are difficult to label and need time and effort from experts, (3) we want to generate

feedback even for the very first student (also known as the cold start problem), and (4) grading requires precision and misgrading has a high cost [8].

However, open response questions are an important way to gauge student understanding and as math curricula continue to encourage 'explain your reasoning' type questions, we see many questions requiring open responses. Unfortunately, teachers often don't grade and/or provide feedback for these types of questions. From our analysis, less than 15% of assigned open response questions in the ASSISTments system receive grades and less than 4% get feedback comments [7]. Figure 1.1 illustrates this and also shows a downward slope as the semester progresses, perhaps indicating that grading these open response problems is time-consuming and tedious.

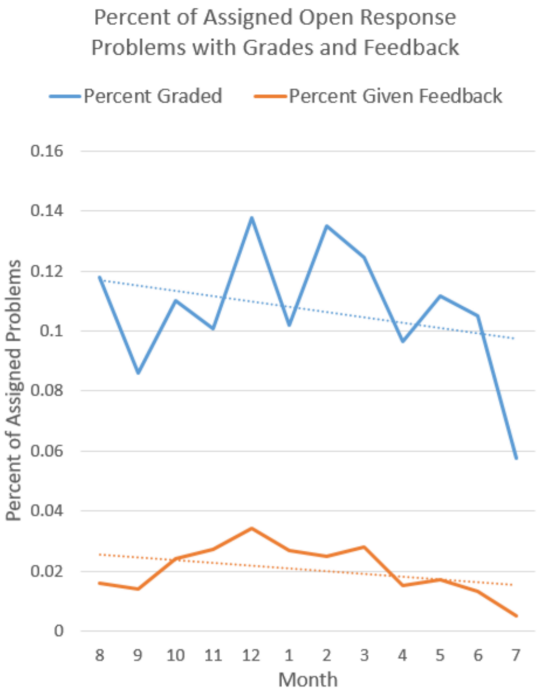


Figure 1.1: Percent of Assigned Open Response Problems with Grades and Feedback

1.1 Background and Related Work

This section describes two research projects currently in development at ASSISTments: QUICK-Comments and LIVE-CHART. The common thread in this thesis work is the development of tools to aid teachers in providing their students with more frequent feedback. These tools are built with the intention of supporting feedback creation in these larger research projects. In QUICK-Comments, the feedback takes the form of natural language message recommendations for open response mathematics questions. In LIVE-CHART, the method of feedback can take many forms, and we create detectors of student behavior to indicate to teachers the kind of feedback that might be appropriate to give to a student at a certain point in time.

1.1.1 QUICK-Comments

With the motivation of wanting to provide teachers better support while grading, and with the knowledge of the difficulties of grading open response questions, the ASSISTments research team began to work on QUICK-Comments. The goal of the QUICK-Comments project is to create a human-in-the-loop AI tool that leverages state of the art NLP to help give feedback for mathematics open response questions [9]. The key is to keep teachers in the driver's seat and provide them with a tool to make the feedback process easier. Figure 1.2 shows the QUICK-Comments interface.

Using the QUICK-Comments interface, a teacher can go through all the student responses submitted for a problem and they receive a suggested numeric score along with three suggested feedback messages for each one. These suggestions are based on how other teachers have evaluated the same problem. Suggestions are never sent to students without the teacher first clicking on it, and modifying it if necessary. Modifying the suggested score will also prompt the tool to change the recommended

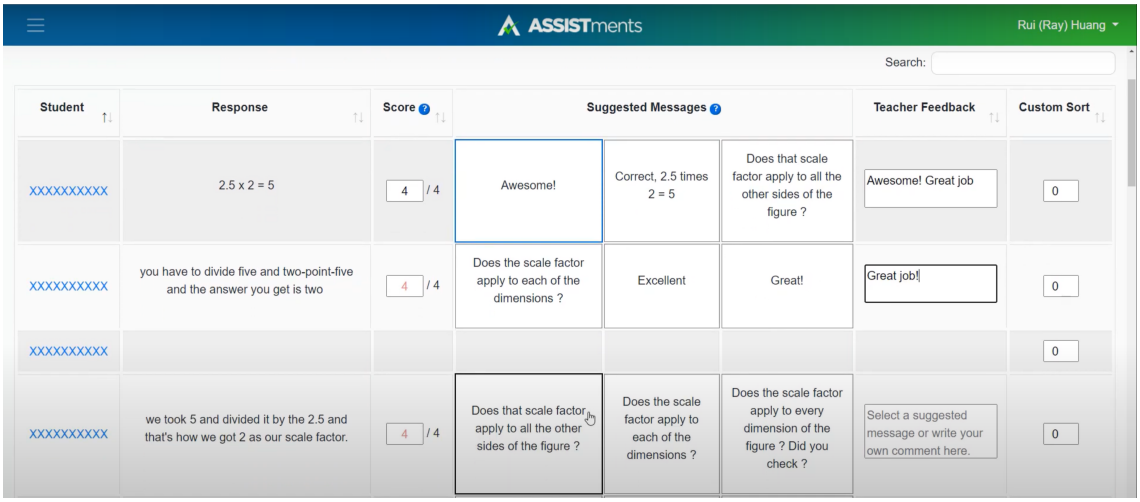


Figure 1.2: QUICK-Comments interface with 3 suggested comments

feedback messages accordingly. For example, if the tool recommends a score of 4, it will also tend to recommend short, positive messages that reflect that the answer deserves full credit. If the teacher decides to change the score to a 3, the QUICK-Comments system will now update with more critical feedback.

1.1.2 LIVE-CHART

LIVE-CHART is a set of real-time awareness tools that will help teachers better manage their students in the classroom. The teacher will be able to see where their students are as they work on a problem set and get notified when students are doing well or need help. From the viewpoint of the student, the tool allows them to discreetly signal to the teacher if they need help without having to physically indicate it.

The tool has straightforward applications in a remote classroom setting, but it could also be beneficial for in-person settings. Just as the Lumilo tool [6] leverages mixed-reality to augment a teacher's ability to pay attention to the students that need the most help, we hope to show a similar result with mobile and web-based

versions of LIVE-CHART. A sample interface for LIVE-CHART can be seen in figure 1.3. In this view, the teacher can see each of their students making their way through a problem set and their performance on the last 5 problems. If the student is struggling, their icon lights up red and they get placed in the 'Requires Attention' list. If the student is progressing fine, their icon lights up green and they are put in the 'Students Doing Well' list.

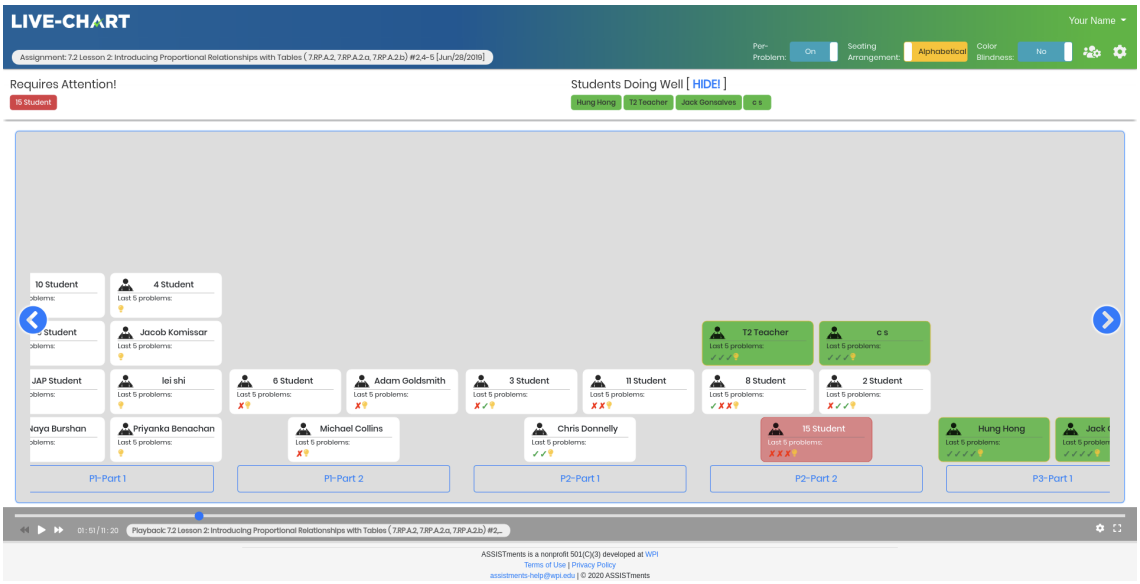


Figure 1.3: Problem view of the LIVE-CHART system

Chapter 2

Evaluating NLP-based Feedback Models

Our previous work has been focused on creating methods and infrastructure to evaluate the performance of feedback systems that are based on natural language generation. The first piece towards achieving this is the creation of evaluation metrics to measure our recommendation performance. The second is the infrastructure to run randomized controlled trials (RCTs).

2.1 Evaluating Comment Suggestions

In order to evaluate a particular recommendation policy, we needed a metric that would represent a good suggestion. To create this, we used a dataset of student answers where multiple teachers graded each student answer. The metric we constructed is a measure of the overlap between different teacher comments for a single answer. We call it the Teacher Agreement Score (TAS).

Taking a particular student answer, we find the top R similar answers using our

recommendation policy. We then do a one-to-one comparison across the T teacher categories for each similar answer and find the average co-occurrence. We then average this value over the R similar answers. Equation 2.1 describes calculating TAS of a single student answer.

$$TAS_i = \frac{1}{R}\sum_{j=1}^{R}\frac{1}{T}\sum_{t=1}^{T} int(Category_{i,t} = Category_{j,t}) \tag{2.1}$$

In the example in image 2.1, $R = 3$ and $T = 4$. We are calculating the TAS for the student answer "no because the 50 and 10 amt the same3" with 3 similar answers and 4 teacher categories.

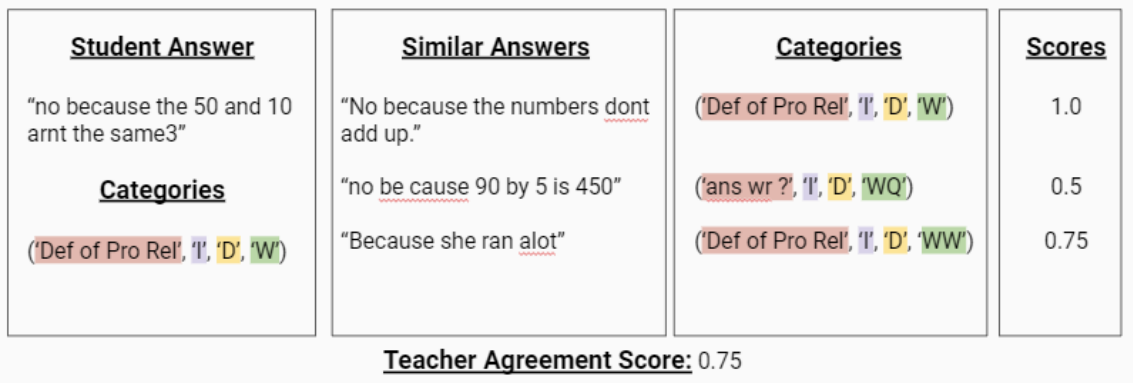


Figure 2.1: Calculating the Teacher Agreement Score (TAS) for a student answer

Once we have calculated the TAS for a particular answer, we go through all the answers and repeat the calculation in a hold-one-out manner (holding out the answer and finding the top R most similar answers). Averaging across all the answers, we get the TAS for a particular problem. Repeating the process across all the problems gives us an overall TAS for the recommendation policy. We use both the per-problem TAS and overall TAS to compare recommendation policies.

This work was done in collaboration with John Erickson, Taylor Stefovic, Priyanka Benachamardi and others, and went into a broader analysis of combinations of sentence representation methods and distance metrics to compare recommendation

policies.

2.2 RCT Infrastructure for QUICK-Comments

Randomized Controlled Trials (RCTs) or A/B Tests are an important method to iteratively improve software. The main idea is to randomly assign users into two groups: control and treatment. The treatment group is shown the new feature being studied while the control group is given the 'business as usual' condition. In order to support RCTs for QUICK-Comments, software components and data tables had to be designed.

The overall flow of data through the QUICK-Comments tool is shown in figure 2.2. The green RCT layer is the new component we designed and it represents the functions and database tables that handle the randomization and data logging. The detailed ERD for the database tables can be found in the appendix.

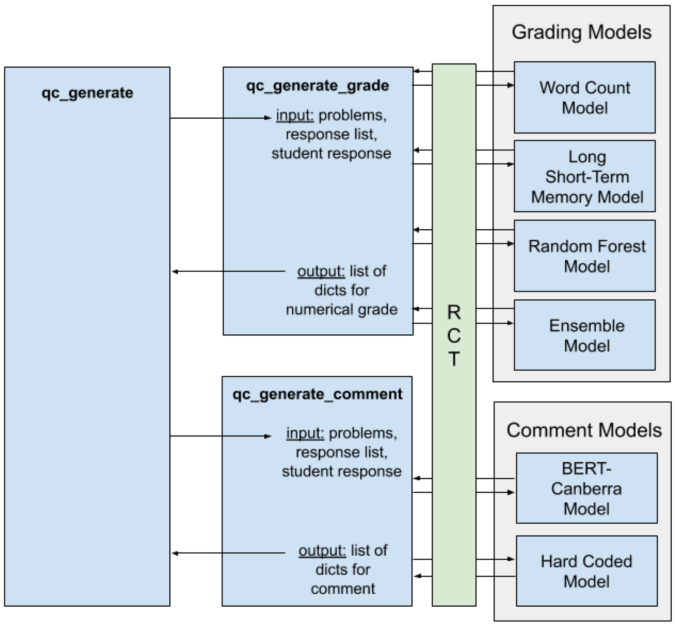


Figure 2.2: RCT infrastructure design for QUICK-Comments

In order to generate feedback comments, the QUICK-Comments tool makes use of 2 kinds of models: grading models and comment models. Since the quality of an answer (as measured by the grade) predicts the kind of comment that is appropriate, we first generate a predicted grade and use that as a feature in our comment generation.

The new RCT layer of the QUICK-Comments tool fits between the function calls to generate comments and the trained models. With this layer, researchers will have the ability to randomize which models are used based on the given user conditions to test out different comment generation strategies. There can be three different levels of user condition randomization. An experiment can be randomized on the teacher, teacher-assignment, or teacher-assignment-student level. These three possibilities for randomization are based on the identification numbers of the user or the assignment.

This work was done in collaboration with Sami Baral and Kirsten Hart and has since been built upon by Priyanka Benachamardi for her thesis work.

Chapter 3

Activity Detectors for Problem-Solving Action Data

The focus of QUICK-Comments is using NLP to generate and suggest feedback messages to students' open responses in mathematics. But there are other areas that we can provide feedback to students by leveraging the data logged as they work on ASSISTments and solve their homework. Feedback can be more than just based on the final answer a student reaches. We believe that there is a lot we can do to help students by providing them feedback on their approach to solving problems as they do their homework.

Our current work expands feedback generation to student actions. The objective of this work is to make it easier for teachers to leave feedback and increase the amount of feedback that students receive. The key differences in this work are the training data and the form that the feedback takes. In this case, what we are training on is student action data - a series of timestamped actions. We are using this data to create detectors of feedback which can be reported to the teacher. The teacher can use the output of these detectors to give feedback to students. If these

detectors are integrated with LIVE-CHART, the feedback takes the form of live conversations or text messages.

3.1 Dataset and pre-processing

This data was collected by showing teachers a snippet of student actions and having them tag the behavior they observed the student exhibiting. Student actions are timestamped interactions that a student makes with the ASSISTments platform as they solve problems (e.g. 'Submitted a response', 'Requested the answer', 'Finished problem'). The teachers select the series of timestamps they are commenting on and leave a comment and comment category. Figure 3.1 shows an example of the screen the teachers saw as we performed this data collection.

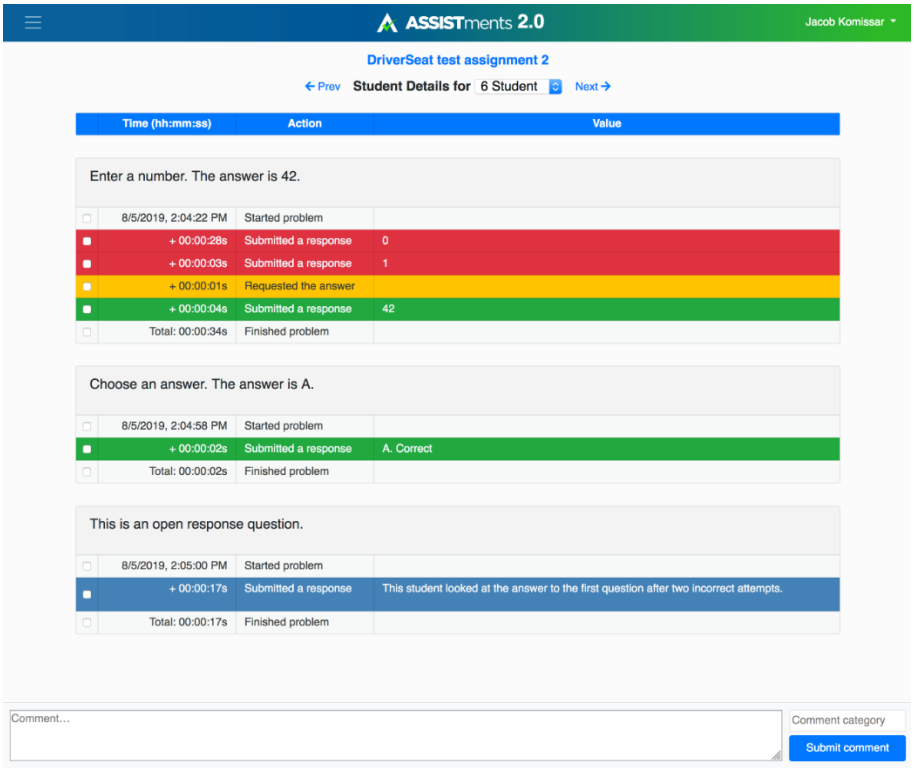


Figure 3.1: Feedback comments for student actions

A significant amount of work for this project went into cleaning the data and pre-processing it to create our model. In addition to this, some pre-processing steps were needed so that we were following previous work and doing a student-level evaluation ensuring that we weren't learning from and predicting on the same student. The following subsections describe the pre-processing steps taken.

3.1.1 Feature Creation

Each action in ASSISTments is given an action name. Some examples of these are 'AnswerRequestedAction', 'HintRequestedAction', 'ProblemFinishedAction'. Each action name was merged with the answer correctness. When the action had no answer correctness, it would be blank, otherwise it would have a true or false appended to the name. For our first model, we had 28 features: 27 features were the one-hot-encoded action name and answer correctness combination, and one feature indicated the seconds since last action.

Some possible features that can be used in the future are the answer text and the problem type. The problem type will need to be found in the ASSISTments database. For the text feature, there will need to be an embedding step. We would expect to see that `answer_text` correlates with answer correctness if we have NLP models trained per problem to learn which phrases score the most points.

3.1.2 Maintaining Student Sequences Within Fold

When testing out our model, we decided to perform cross-validation. Following similar work on affect detection [4][5], we wanted to distribute students across the folds to create 10 roughly equal-sized groups and perform a student-level cross validation. Due to the nature of the data where different students have different action sequences, this means that the data within folds won't be exactly equal - but there

won't be large disparities when looking at large numbers of students.

3.1.3 Collapsing Categories

Since teachers were free to leave their own comments and create their own comment categories as they saw fit, there is overlap between categories of different teachers with slight differences. One of the things we explored was ways to collapse these categories in order to create labels that we could predict across teachers.

We took a few approaches to this, ranging from hand-coding the categories to exploring word embeddings. However, these techniques are reliant on semantic similarity and since we did not have a measure of inter-rater reliability, we could not use it. In order to collapse categories effectively, we would need to have multiple teachers grade the same responses so that we have a measure of agreement across teachers. Once we have a model that is built for predictions across teachers, we can use the embedding layer to find which categories overlap in an unsupervised manner (such as clustering). This would give us a method of condensing down the categories in a way that does not rely on semantics.

3.2 First Model: When to give Feedback?

The first model created, takes in a series of student actions and outputs a binary prediction of which actions the teacher left feedback for. Essentially, the model attempts to flag which actions taken by a student are noteworthy.

Since LSTMs are good models for time series and useful for modelling sequences that may have lags between timesteps that are significant, we thought it would be suitable to apply here. When teachers select a sequence of actions to comment on, we do not know which parts of the sequence contain information that is more

significant. We hope that in the aggregate, this information can be learned.

We implemented the lstm model using the Keras API for TensorFlow. We created a sequential model that had an lstm layer with 50 hidden unites and a dense output layer with sigmoid activation. The model was trained for 200 epochs with early stopping and a patience of 10. This usually caused training to stop after 30 epochs, and plotting the validation loss indicated that this was around optimal. As described earlier, the model was fit and evaluated 10 times to do a 10-fold cross validation.

All code for this work can be found at github.com/aaron-alphonsus/student-actions. This repo only contains code - any data that was used will have to be requested from the ASSISTments research team: www.assistmentstestbed.org/faq.

3.3 Second Model: Detecting Positive Feedback

In our second model, we had two goals we wanted to meet. The first was that we wanted to be able to create something that could be applied to the infrastructure already in ASSISTments. LIVE-CHART was identified as a potential candidate for this. With that in mind, we began to look at the data to pick out a potential behavior that we thought would be interesting to detect. We also needed to keep in mind that there needed to be a decent number of these instances present.

From the literature, we noticed that studying positive feedback could be an interesting angle to approach this as it wasn't focused on as much as critical feedback. Looking at the data, we picked out the teacher that had graded the most problems and chose one of their categories as our behavior of interest. This ended up being the category 'Thoughtful self correcting'. In all, this teacher had 27 different categories. 1177 actions were flagged and marked with 'Thoughtful self correcting'. The vast majority of the actions were not commented on (150430 actions)

Once again, our implementation was an lstm model using Keras. The architecture and training method was the same as our previous approach. This time however, the model tended to train for 70-80 epochs before early stopping. The evaluation of the model was done the same way, using a student-level 10-fold cross validation.

3.4 Results

Both models were evaluated by performing 10-fold cross validation, being careful to maintain student sequences within a fold. AUC and RMSE were calculated for each fold and averaged across folds. The first model achieved an AUC = 0.602 and RMSE = 0.298. The results of the second model that detects positive behavior was slightly better, achieving an AUC = 0.645 and RMSE = 0.161. To contextualize these results, we look to affect detection and the work done by Ocumpaugh et al. [4] and Wang et al. [5]. These were some of the first attempts to create detectors of affect, and the AUC achieved is comparable. However, both of these works use many more features than we do (69 and 232 respectively) and feature engineering could be an area to improve on in our work (e.g. a feature indicating the type of question - true/false, multiple choice, open text)

Chapter 4

Discussion

In this work we extend our work on feedback to a dataset of student actions. We created models for the sequence to sequence tasks of 1) detecting when teachers give feedback and 2) detecting a particular positive behavior within the student actions. These detectors can be applied to student reports to flag rows of student actions where students could receive feedback. Since this data was collected from other expert teachers, our predictions are an indication of what a teacher would find notable for comment.

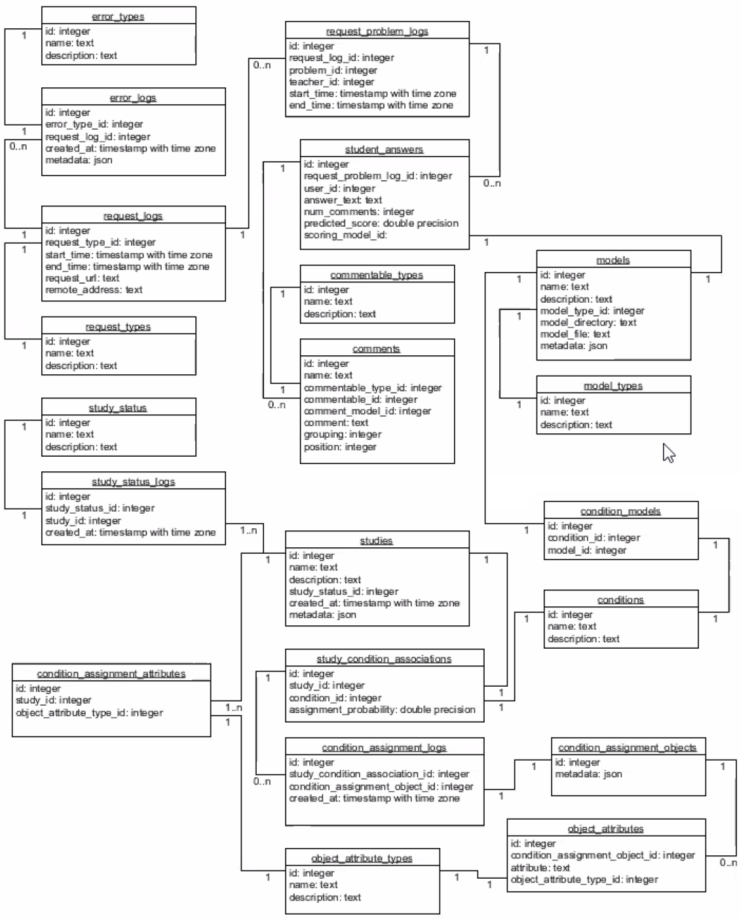
Our next steps would be to examine how categories overlap across teachers so that we can not only predict where we can give feedback but also what category of feedback. In order to do this, we need to find some measure of agreement across teacher categories. This was a simpler task in our previous work as multiple teachers had graded a single problem set so we could just look at a particular student answer and count the co-occurrence of categories from different teachers. In this case, since teachers only graded their own students, we need to come up with a different way to cluster and collapse categories so that we can predict them.

By creating the detector of positive feedback, we have the goal of detecting when

student actions exhibit positive behavior. This detector can be used to analyze simple timestamped action data to detect the presence of the behavior 'Thoughtful self correcting'. With tweaks to the model, we can build towards a detector that could be deployed for a real-time setting like LIVE-CHART. To do this, we will need to modify the output so that it makes a prediction only for the final timestep in the sequence, instead of a prediction for the entire sequence.

Appendix A

ERD for QUICK-Comments RCTs



Bibliography

[1] A. D. Rowe, L. N. Wood, *et al.*, "Student perceptions and preferences for feedback," 2008.

[2] S. A. Schartel, "Giving feedback – an integral part of education," *Best Practice & Research Clinical Anaesthesiology*, vol. 26, no. 1, pp. 77–87, 2012. DOI: 10.1016/j.bpa.2012.02.003.

[3] P. Kehrer, K. Kelly, and N. Heffernan, "Does immediate feedback while doing homework improve learning?" In *FLAIRS Conference*, 2013.

[4] J. Ocumpaugh, R. Baker, S. Gowda, N. Heffernan, and C. Heffernan, "Population validity for educational data mining models: A case study in affect detection," *British Journal of Educational Technology*, vol. 45, no. 3, pp. 487–501, 2014. DOI: https://doi.org/10.1111/bjet.12156. eprint: https://bera-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.12156. [Online]. Available: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12156.

[5] Y. Wang, N. Heffernan, and C. Heffernan, "Towards better affect detectors: Effect of missing skills, class features and common wrong answers," *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, 2015.

[6] K. Holstein, B. McLaren, and V. Aleven, "Student learning benefits of a mixed-reality teacher awareness tool in ai-enhanced classrooms," in *AIED*, 2018.

[7] J. A. Erickson, A. F. Botelho, S. McAteer, A. Varatharaj, and N. T. Heffernan, "The automated grading of student open responses in mathematics," in *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, ser. LAK '20, Frankfurt, Germany: Association for Computing Machinery, 2020, pp. 615–624, ISBN: 9781450377126. DOI: 10.1145/3375462.3375523. [Online]. Available: https://doi.org/10.1145/3375462.3375523.

[8] A. Malik, M. Wu, V. Vasavada, J. Song, M. Coots, J. Mitchell, N. Goodman, and C. Piech, *Generative grading: Near human-level accuracy for automated feedback on richly structured problems*, 2021. arXiv: 1905.09916 [cs.LG].

[9] N. Heffernan. [Online]. Available: https://www.neilheffernan.net/projects/funded-projects/quickcomments.