Improving Feedback Recommendation in Intelligent Tutoring Systems using NLP

by

Priyanka Benachamardi

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 Yanhua Li, Thesis Reader

Professor Craig E. Wills, Head of Department

Abstract

The fundamental principle behind item-based recommender systems is to compare the similarity of an item to a ranked set of items. Our study explores the use of such recommendation system to suggest feedback comments for open-ended student answers in Intelligent Tutoring Systems. Several research works in the past have contributed to the development of technologies and strategies to assess and grade students' work. The study in this paper aims to leverage Natural Language Processing, Machine Learning, and Statistical Methods to build a feedback infrastructure to help teachers in assessing their students' open-ended answers. We investigate multiple approaches in determining the semantic similarity between two answers and evaluate the quality of semantic relatedness using our own metric called Teacher Agreement Score (TAS). It is often considered difficult task to assess open-ended data such as natural language. To evaluate the quality of the system, we have built a Software Infrastructure that enables running Randomized Control Trials on ASSISTments Platform to study the behavior by extracting information on the usage and effectiveness of the developed system.

Acknowledgements

I would like to thank my advisor, Professor Neil T. Heffernan for the guidance I received from him during the last two years. I would like to thank my mentor, Anthony Botelho for his guidance over the Thesis work. His patience and attention to details has been instrumental in making me a better researcher. I would like to thank John Erickson for his support while working on the recommendation model of the thesis. I also want to thank my thesis reader, Professor Yanhua Li, for reading my thesis and giving useful feedback to make this thesis a success.

Contents

1 Introduction **1**

1.1 Goal of the Thesis . . . 1

1.2 Research Contributions . . . 2

2 Background **3**

2.1 Related Work . . . 3

2.2 ASSISTments Quick-Comments . . . 4

3 Feedback Recommendation **5**

3.1 Sentence Similarity . . . 5

3.2 Dataset . . . 6

3.3 Sentence Encoders . . . 8

3.3.1 Sentence-BERT . . . 8

3.3.2 Universal Sentence Encoder . . . 8

3.4 Distance Metrics . . . 8

3.5 Method I - Comparing Encoder-Distance Combinations . . . 9

3.5.1 Finding Similar Answers . . . 9

3.5.2 Evaluation - Teacher Agreement Score . . . 10

3.5.3 Algorithm to compute TAS . . . 12

3.5.4 Experiment Analysis . . . 13

3.5.5 Results 13
3.6 Method II - Ensemble model of Encoder-Distance Combinations . . . 16
3.6.1 Approach to Assemble Dataset 16
3.6.2 Linear Regression Results 17
3.7 BERT-Canberra Model in ASSISTments 18
3.7.1 Current Implementation 18
3.7.2 Pilot Observational Data Analysis 19

4 RCT Logging Infrastructure 21

4.1 Randomized Control Trials 21
4.2 QUICK-Comments RCT 21
4.2.1 Database Design 23
4.2.2 Randomizing User Conditions 25
4.3 BERT-Canberra-Wise Model 27

5 Limitations and Future Work 28

6 Conclusion 29

7 References 30

List of Figures

3.1 Dataset Collection sample 6

3.2 Embedding representations in a 2D vector space 10

3.3 TAS calculation at answer-level 11

3.4 Pseudo code to compute average TAS score 12

3.5 Ensemble of best approaches for Training Linear Regression 16

3.6 ASSISTments Quick Comments with Sentence-BERT-Canberra . . . 18

4.1 Quick Comments RCT High-Level Design 22

4.2 ER Diagram of RCT 24

4.3 Randomization Sequence of Events 26

4.4 Example of Wise suggestions 27

List of Tables

3.1 Encoder-Distance Comparison Report 14
3.2 Encoder-Distance Comparison Reports with Max Possible Score . . . 15
3.3 Number of times obtained TAS agreed with Teachers 15
3.4 Linear Regression Results 17
3.5 Encoder-Distance Comparison Reports with max possible score . . . 19

Chapter 1

Introduction

Intelligent Tutoring Systems (ITS) leverage Artificial Intelligence and Cognitive Science to provide learners personalized instructions to suit their specific requirements[1]. Several studies show Intelligent Tutoring Systems have been proven to be beneficial in the advancement of student learning in the context of education[2]. The technology and research in computer-aided education are constantly evolving to provide autonomous feedback and engage students in active learning.

While many intelligent systems do support immediate feedback, they mostly involve question types such as multiple choice or fill-in-the-blanks or questions with a finite set of answers. This lack or little support for descriptive answers in intelligent systems necessitates the intervention of human teachers.

1.1 Goal of the Thesis

The goal of this study is to delve into one such research aspect and develop an NLP model that auto-suggests feedback comments to student answers based on item-based recommendation and then incorporate the model into ASSISTments' QUICK-Comments feature. In this work, we provide a method to recommend feedback comments utilizing all the previously seen teacher comments for a given problem. The assumption that we make here is, if a student's expertise on the subject is at a certain level, then, other students at the same level of expertise, on the same subject, in general, may have a similar understanding of the subject and are likely to make similar mistakes or have similar misconceptions. Thus, paving a way for recommendation systems to cluster a pool of similar answers and recommend appropriate feedback comments.

We attempt to improve the relevancy of feedback comments by taking into account the syntactic and semantic similarity when comparing two answers and explore new ways of formulating feedback comments. We also build infrastructure to support exploration beyond the scope of this thesis by enabling the system to apply randomization policies to the suggested feedback comments and support logging user actions.

1.2 Research Contributions

In this study, we make the following contributions towards NLP-based text recommendation.

1. We formulate a method to recommend teacher comments to a new student answer, by making use of historic feedback samples of a given problem.

2. We devise an evaluation metric that measures the recommendations proposed by the model.

3. We build a software infrastructure to support running RCTs within the QUICK-Comments system.

Chapter 2

Background

2.1 Related Work

Prior research works have attempted to grade and provide feedback to open-ended answers by majorly using a corpus-based approach, or by modeling a set of correct answers using structured domain knowledge. C-Rater[4] uses predicate-argument structure, pronominal reference, morphological analysis, and synonyms to provide full or partial credit to an open-ended answer.

A link grammar[5] based approach uses the extraction of the keywords and then identifies and extracts the relational expressions for that keyword, the model utilizes the relational expressions to automatically evaluate student's short answer with respect to a modeled answer.

Earth Mover's Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading (ASAG) [6] provides a framework for grading open-ended student answers. It utilizes an attention mechanism with Siamese bidirectional LSTM trained onto a model and a student answer.

Google's Smart Reply[7], provides an end-to-end framework for automatically generating short email responses. The LSTM model trained on tech support chat corpus generates semantically diverse suggestions that can be used as complete email responses.

2.2 ASSISTments Quick-Comments

ASSISTments hosts a platform for teachers to grade and provide feedback on student assignments. The students solve assigned math problems, provide open-ended explanations to their reasoning, and submit answers asynchronously. The teacher then accesses the submission to grade and provide feedback. The system currently can auto-grade the submitted answer by assigning a numerical grade ranging from 0-4. The system also suggests 3 hard-coded simple but useful feedback comments. The teachers can pick one of the suggested comments or they can choose to write their own. Our study explores ways to suggest problem-specific feedback comments to a student answer. We utilize this platform to run our experiments and study the behavior.

Chapter 3

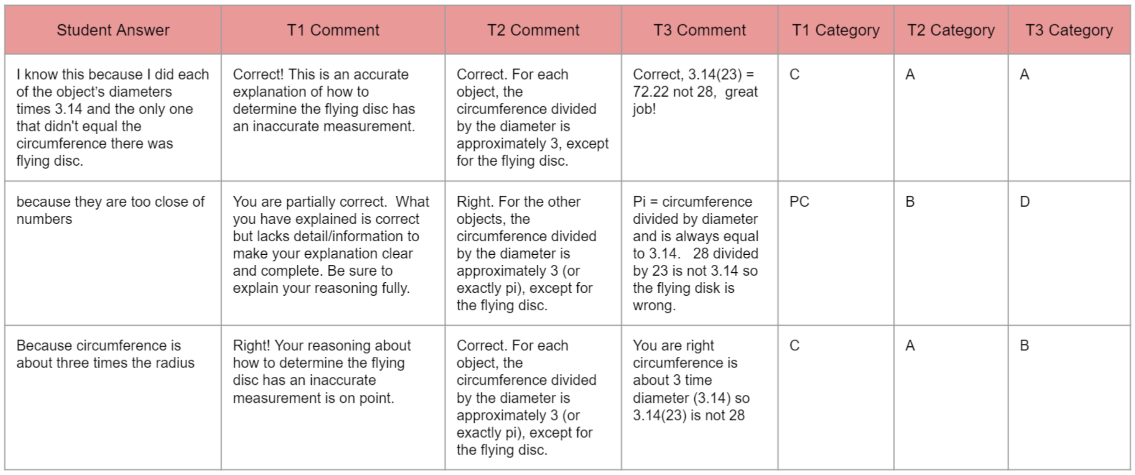
Feedback Recommendation

3.1 Sentence Similarity

The similarity between two sentences is determined by comparing both lexical and semantic closeness of the sentences. One of the many ways to identify the similarity between two documents is to take an average of the number of common characters they share or compute a cosine distance between them. Methods such as TF-IDF, Bag of Words, Word2Vec, and Glove pre-trained encoder do not take the word order into account. The differences in the order of the words often change the meaning behind it. It is vital to parse full text to capture the meaning of the student's answer, and hence, this study focuses on the syntactic and semantic similarity when making teacher response recommendations. We explore pre-trained NLP sentence encoders namely Sentence-BERT[8] and Universal Sentence Encoder (USE)[9] to transform the student answers into sentence embeddings in a vector space.

3.2 Dataset

Seventeen middle school mathematics teachers were recruited for the data collection task. They used Open Educational Resources (OERs) in their classrooms to assign, score, and provide feedback for open-response problems. The Dataset is a collection of CSV files, each of which corresponds to a particular problem ID. Each problem is graded by 1-4 teachers and contains an average of about 80 student answers. Each CSV file contains the following attributes: student ID, the problem ID of a specific problem the student had worked on, the student's open-ended answer to the problem, the teacher comments(up to 4 teachers), and the teacher comment categories(up to 4 teachers). The teacher comments are pooled in from various teachers grading the same student answer. Teachers were asked to categorize the comments into categories using their desired naming, and desired number of categories. They were given no further instructions on how to categorize the comments. This was done in attempt to capture the essence of how teachers think of similar comments.



Student Answer	T1 Comment	T2 Comment	T3 Comment	T1 Category	T2 Category	T3 Category
I know this because I did each of the object's diameters times 3.14 and the only one that didn't equal the circumference there was flying disc.	Correct! This is an accurate explanation of how to determine the flying disc has an inaccurate measurement.	Correct. For each object, the circumference divided by the diameter is approximately 3, except for the flying disc.	Correct, 3.14(23) = 72.22 not 28, great job!	C	A	A
because they are too close of numbers	You are partially correct. What you have explained is correct but lacks detail/information to make your explanation clear and complete. Be sure to explain your reasoning fully.	Right. For the other objects, the circumference divided by the diameter is approximately 3 (or exactly pi), except for the flying disc.	Pi = circumference divided by diameter and is always equal to 3.14. 28 divided by 23 is not 3.14 so the flying disk is wrong.	PC	B	D
Because circumference is about three times the radius	Right! Your reasoning about how to determine the flying disc has an inaccurate measurement is on point.	Correct. For each object, the circumference divided by the diameter is approximately 3 (or exactly pi), except for the flying disc.	You are right circumference is about 3 time diameter (3.14) so 3.14(23) is not 28	C	A	B

Figure 3.1: Dataset Collection sample

In the figure 3.1, we see a student's open-ended answer, three teacher comments belonging to three different teachers, and three teacher category columns each of which corresponds to the teacher comment. The teacher categories are labeled in such a way that similar comments are grouped under a single category. i.e, a teacher with their own judgement, groups a bunch of their comments into one category, indicating that all the comments within that group are similar comments. Teachers were intentionally not provided with any additional info on how to group the comments, with the intent to capture variations in how teachers think about the comment similarity. When two different teacher comments in the same teacher column are assigned the same category label in the corresponding teacher category, we assume that the comments are similar in meaning. i.e, they are similar in the eyes of the teacher which can be used by our model to suggest as a similar answer candidate. For example the 1st and 3rd comment in Teacher 1 column are different but are labelled as C in teacher 1 column, meaning they are similar. The category coding of one teacher means entirely different when comparing the same coding for another teacher category in another column. The category 'A' in teacher 2 column is note the same as category 'A' in teacher 3 column. The category labels are unique to individual teachers. We will be making use of these labels in determining how a teacher think of similar comments with respect to other teachers.

As a pre-processing step, we discarded problems with zero teacher responses, eliminated rows with empty student answers, and removed rows that had less than 1 matching categories. i.e. if a teacher has categorized an answer say 'X' as 'AF' and there is no other answer in the same teacher column that shares the same category then it means that the dataset does not contain a similar answer entity for the answer 'X'. Such samples are of less significance to us since we are attempting to establish a metric that determines what "similar" means to the teacher audience.

3.3 Sentence Encoders

3.3.1 Sentence-BERT

Sentence-BERT is a deep learning NLP model tuned for efficient performance on the task of semantic textual similarity (STS). Sentence-BERT is modeled after BERT[10] which was able to achieve state-of-the-art performance on STS. However, finding the most similar pair in a corpus of 10000 sentences takes around 65 hours with BERT. Sentence-BERT modifies the BERT construction by using Siamese and triplet network structures which cuts down that time to 5 seconds. This makes it ideal for use in our work.

3.3.2 Universal Sentence Encoder

Universal Sentence Encoder (USE) is another pre-trained embedder by Google which is trained on a large wiki corpus with supervised and unsupervised tasks and encodes sentences into a 512d vector space. USE uses the transformer architecture and provides capability to look at text context when generating embeddings for entire sentences.

3.4 Distance Metrics

We choose 4 distance metrics namely Levenshtein Distance, Euclidean distance, Cosine distance, and Canberra distance to include a wide variety of metrics. We pick these both for their large usage in NLP tasks and significant difference in their meaning.

The Levenshtein distance is the minimum number of the edits required to transform one sentence into another[11]. Levenshtein distance is applied on raw text and not vector representations of the sentence.

The Euclidean distance between two points measures the geometric length between two points in Euclidean space and is computed as the summation of squared differences between two points.

Cosine distance is the cosine of the angle between two n-dimensional vectors in an n-dimensional space. It is the dot product of the two vectors divided by the product of the two vectors' lengths. Cosine distance is widely used in applications like plagiarism checker.

Canberra distance is another numerical measure of the distance between pairs of points in a vector space and has been used as a metric for comparing ranked lists[12] Canberra distance is similar to the Manhattan distance. The only change is that the absolute difference between the variables of the two points is divided by the sum of the absolute point values prior to the summation.

3.5 Method I - Comparing Encoder-Distance Combinations

3.5.1 Finding Similar Answers

We start by transforming the student answer corpus to their equivalent embedded vectors, then we use the vectors to compute the pair-wise distance between a holdout answer and the rest of the answers in that problem set. This gives us an array of distances between each pair. We then sort the array to get the top 3 distances. In other words, we take the top 3 closest answers to the holdout answer. The reasoning behind computing distances is that the answers in vector space appear closer to answers that are both syntactically and semantically similar.

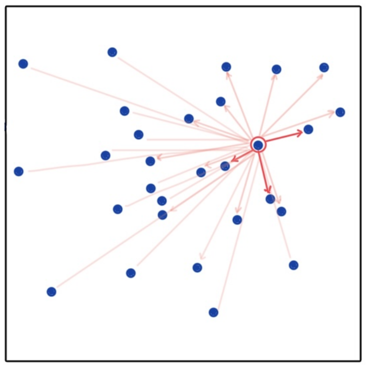


Figure 3.2: Embedding representations in a 2D vector space

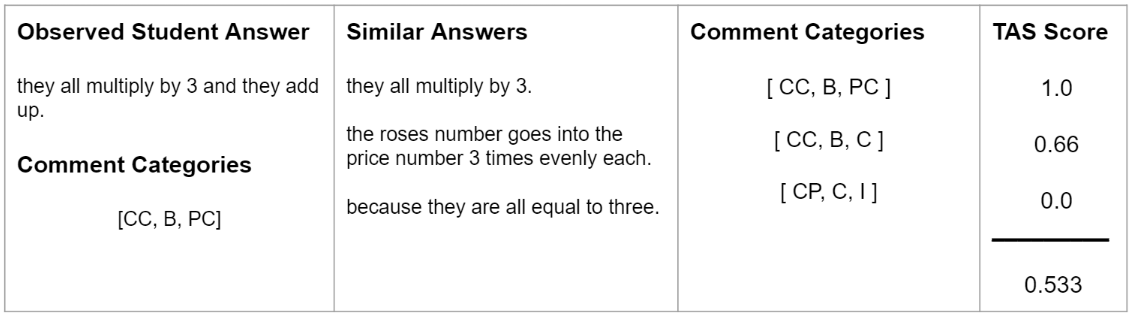
3.5.2 Evaluation - Teacher Agreement Score

The above method to compute similar answers regardless of the choice of encoder or distance metric, will always produce some closest answer which we consider as the similar answer and they all might look good to the naked eye. But How do we know which one are the most similar?

We evaluate how good the pre-trained model's suggestions are by calculating how much category label overlap there is between observed student answer and similar answers found. This is done through a simple average co-occurrence measure that we call Teacher Agreement Score (TAS). TAS is the count of the number of co-occurring categories divided by the total number of categories.

As mentioned earlier, we asked the teachers to categorize or label the feedback comments into desired number of groupings. These are labels are used to define a measure representing similarity as defined by the teachers cohort. These category labels act as the ground truth values to compare our model category recommendations. The TAS can range from 0-1 based on the number of matching categories. A higher TAS value indicates more likeliness for the suggested comments to be utilized by teachers.

For the top 3 similar answers found, we retrieve the associated teacher categories



Observed Student Answer	Similar Answers	Comment Categories	TAS Score
they all multiply by 3 and they add up.	they all multiply by 3.	[CC, B, PC]	1.0
Comment Categories	the roses number goes into the price number 3 times evenly each.	[CC, B, C]	0.66
[CC, B, PC]	because they are all equal to three.	[CP, C, I]	0.0
			0.533

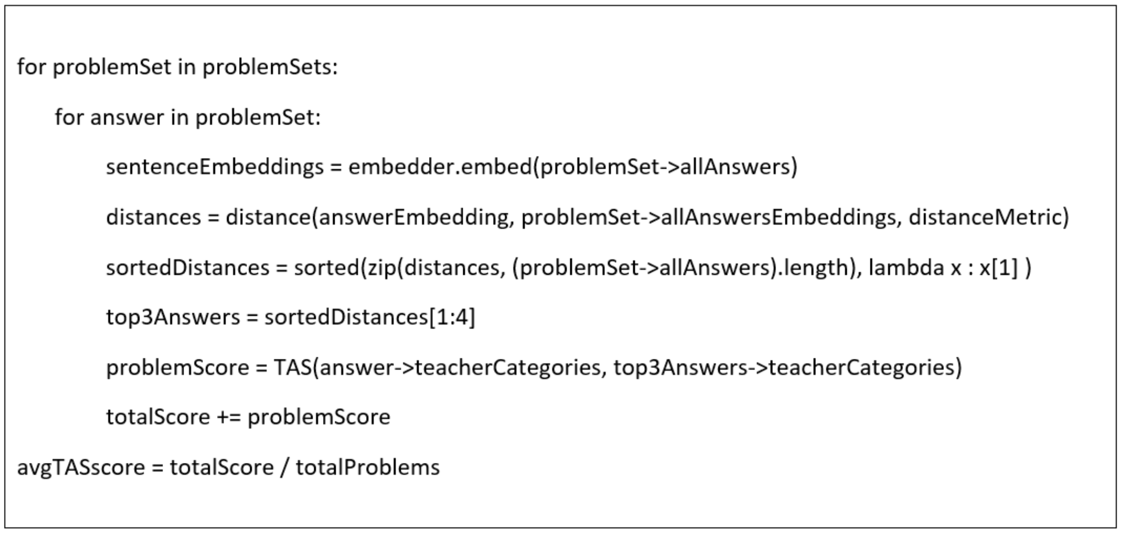
Figure 3.3: TAS calculation at answer-level

and do a 1-1 comparison and average the score by the number of teacher categories T. This process is then repeated in a hold-one-out manner observing each student answer as the selected holdout and an average TAS is calculated for the similar answers found with respect to the given problem. Finally, this process is repeated across all the problems and an average TAS and per-problem TAS is reported.

$$TAS_i = \sum_{j=1}^{3} \frac{1}{T} \sum_{t=0}^{T} int(Category_i = Category_j) \tag{3.1}$$

In the example shown in fig 3.3, the observed student answer has 3 categories CC, B, and PC. Again the naming conventions have no meaning, it is just what teachers chose. The first similar answer has categories CC, B and PC, which is a complete overlap with a total 3 and divided by number of categories gives a TAS score of 1. For the second we see a TAS score of 0.66 since two of the categories overlap. And for the third none of them overlap resulting in a TAS score of 0. We finally average the 3 TAS scores of similar answers and this gives us a TAS score for the observed student answer. Which is 0.533. TAS metric demonstrates to be a good evaluation metric since it captures the similarity measure according to what the teachers think is similar. i.e, the dataset contains the ground truth values which are categorized by a group of teachers. We are essentially modelling the system to think like teachers.

3.5.3 Algorithm to compute TAS



```
for problemSet in problemSets:
    for answer in problemSet:
        sentenceEmbeddings = embedder.embed(problemSet->allAnswers)
        distances = distance(answerEmbedding, problemSet->allAnswersEmbeddings, distanceMetric)
        sortedDistances = sorted(zip(distances, (problemSet->allAnswers).length), lambda x : x[1] )
        top3Answers = sortedDistances[1:4]
        problemScore = TAS(answer->teacherCategories, top3Answers->teacherCategories)
        totalScore += problemScore
avgTASscore = totalScore / totalProblems
```

Figure 3.4: Pseudo code to compute average TAS score

This algorithm represents finding similar answers and computing TAS for the entire dataset. We calculate dataset TAS to analyze which approach performs the best on overall data. For each problemSet say ‘p’ i.e a csv file containing student and teacher data for a specific problem ID in all the problemSets ‘P’, and for each student answer ‘a’ in each problemSet ‘p’, we get the sentence embeddings for all the answers in current problem ’p’ using a chosen pre-trained encoder. Next, we calculate the distance between current holdout answer embedding and other answers’ embeddings using the chosen distance metric. We sort the distances in the order that best defines higher similarity and take the top 3 distances. Now we have our Top 3 similar answers pairs.

To Evaluate the similar answer pairs, We get the list of teacher categories for the observed answer ’a’ and the lists of teacher categories for all the similar answers found. Then we calculate TAS for an answer 'a' as described in the equation (3.1) above which is the answer level TAS score. All the answer level TAS scores are averaged to get a problem level TAS score. And then finally, all the problem level TAS scores are averaged to get a dataset TAS score.

3.5.4 Experiment Analysis

In this experiment, we analyze the overall TAS performance of different approaches of encoder-distance metric combinations on entire Dataset. As a baseline model, we simply compute the Levenshtein ratio between the textual answer pairs. Levenshtein distance does not account for varying lengths. Therefore, we simply compute Levenshtein ratio between a pair of answers. Levenshtein ratio computes levenshtein distance and divides by the character alignment length to account for varying length in the sentences. The pre-trained encoders Sentence-BERT and USE are chosen for their popularity in the NLP world and for being the current state-of-art sentence encoders. We compare the metrics Euclidean, Cosine and Canberra in combination with both the sentence encoders.

3.5.5 Results

In the table 3.1, we show the TAS scores obtained for each of the encoder-distance combinations. Levenshtein Ratio has a TAS of 0.536 which we use as the baseline for our model. USE-Euclidean and USE-Cosine have slightly increased score compared to Levenshtein which is at 0.556 and the USE score slightly reduces with Canberra at 0.554. Sentence-BERT has shown to perform the best among the models that we have considered with 0.623 for Cosine and Canberra.

It is possible that for any given problem set, the maximum possible TAS score(MPS) may always be less than 100 percent. i.e, in any given problem set, if there are no

Encoder-Distance	**TAS score**
Levenshtein Ratio (No Encoder)	0.536
Universal Sentence Encoder-Euclidean	0.556
Universal Sentence Encoder-Cosine	0.556
Universal Sentence Encoder-Canberra	0.554
Sentence BERT-Euclidean	0.621
Sentence BERT-Cosine	0.623
Sentence BERT-Canberra	0.623

Table 3.1: Encoder-Distance Comparison Report

two similar comments such that their categories have a complete overlap, the maximum possible TAS will always be less than 1. In such cases, if we average it by the total number of categories then we're penalizing the TAS score. To mitigate this penalty, we also keep a record of MPS for each problem. Now, we apply the same evaluation algorithm as above to compute the average TAS score for the entire dataset by taking the top 3 predictions for each answer. This time we also divide the per-problem TAS by the MPS to boost the confidence of the model. Note that as a pre-processing step we already discarded, answers that did not have at least one similar answer candidate with at least one overlapping teacher category. So, the Max Possible Score will never be zero. The results are reported in table 3.2 below.

$$TAS_i = \sum_{j=1}^{3} \frac{1}{T} \sum_{t=0}^{T} (int(C_i = C_j))/MPS_i \tag{3.2}$$

Averaging the TAS by MPS slightly increased the confidence in the model. We see the best performing Sentence-BERT TAS increased by nearly 0.2 units, which is not a large enough difference but still an improvement. For all future experiments, we will use the revised TAS calculation formula(3.2). This step does not make a difference in finding the best approach, but just a better metric overall.

It's no doubt that SBERT performed the best among the comparison approaches.

Encoder-Distance	**TAS score**
Levenshtein Ratio (No Encoder)	0.559
Universal Sentence Encoder-Euclidean	0.581
Universal Sentence Encoder-Cosine	0.581
Universal Sentence Encoder-Canberra	0.576
Sentence BERT-Euclidean	0.649
Sentence BERT-Cosine	0.651
Sentence BERT-Canberra	0.651

Table 3.2: Encoder-Distance Comparison Reports with Max Possible Score

Encoder-Distance	**Number of times Best TAS**	**% Best TAS**
Levenshtein Ratio (No Encoder)	1/67	1.49%
USE-Euclidean	8/67	11.94%
USE-Cosine	8/67	11.94%
USE-Canberra	3/67	4.48%
Sentence-BERT-Euclidean	12/67	17.91%
Sentence-BERT-Cosine	17/67	25.37%
Sentence-BERT-Canberra	27/67	40.3%

Table 3.3: Number of times obtained TAS agreed with Teachers

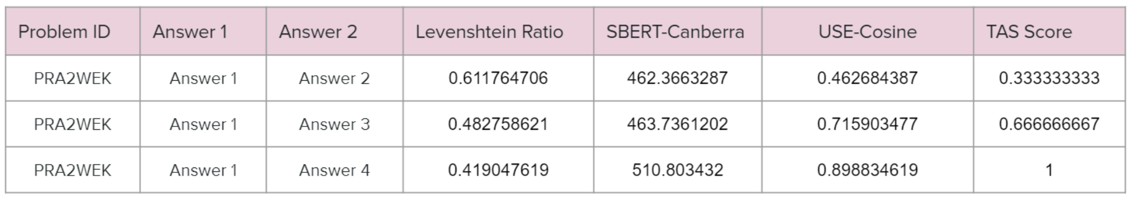
But with respect to our dataset, none of them either performed too poorly or too strongly. Also the choice of the distance is an important factor in improving the suggestions. To study how these approaches performed against each other, we compute problem wise performance . We compared each of the approaches to see how many times they agreed with teachers. Table 3.3 provides the number of times an approach had the best possible TAS score for a given problem. Out of the 67 problems Levenshtein Ratio agreed with teachers at least once. USE-Euclidean and cosine managed to top the TAS for 8 problems each. USE-Canberra seemed to have managed only 3 times. A clear indication that this combination may not be useful in our model. Sentence-BERT with consistent high TAS scores appears to have topped

12 times for Euclidean, 17 times for Cosine, and 27 times for Canberra. There were 9 cases where at least two approaches had the same top score. We deemed both approaches to be counted as the best performing approach. Sentence-Bert with Canberra has managed to agree at least 40.3% of the time, making it the winning approach among the Encoder-distance combinations considered in this work. From this data we can infer that there is no single policy that outperforms others in every case, but SBERT-Canberra is arguably the best achieving 40.3% Best TAS score.

3.6 Method II - Ensemble model of Encoder-Distance Combinations

3.6.1 Approach to Assemble Dataset

Since each encoder-distance approach performed the best on certain problems, With ensemble model, we try to bring the best of each encoder and train a linear regression to predict TAS scores between answer pairs. We assemble a dataset by combining Levenshtein ratio, Sentence-BERT-Canberra distance, and USE-Cosine distances between the answer pair as independent variables and the TAS score of the answer pair being the dependent variable or the ground truth value.



Problem ID	Answer 1	Answer 2	Levenshtein Ratio	SBERT-Canberra	USE-Cosine	TAS Score
PRA2WEK	Answer 1	Answer 2	0.611764706	462.3663287	0.462684387	0.333333333
PRA2WEK	Answer 1	Answer 3	0.482758621	463.7361202	0.715903477	0.666666667
PRA2WEK	Answer 1	Answer 4	0.419047619	510.803432	0.898834619	1

Figure 3.5: Ensemble of best approaches for Training Linear Regression

The figure 3.5 shows a sample of assembled dataset. We pick a character comparison approach Levenshtein Distance to capture the textual similarity aspect between the sentences. USE-Cosine and USE-Euclidean both were a tie in terms of overall TAS score, but we pick USE-cosine for popularity in NLP similarity tasks. SBERT-Canberra is chosen as it was the best performing approach. We compute these distances between every answer pair of every problem set and use the TAS score between them to be the ground truth value for training the model.

3.6.2 Linear Regression Results

We trained a Linear regression model with a hold-one-answer-out policy. The predicted TAS for the held-out answer pairs are recorded. For each held-out answer, we sorted the predicted TAS scores in increasing order and took the top 3 answer pair predictions as the most similar answers. We use the TAS evaluation from the Algorithm above to calculate the average TAS for the entire dataset.

Root Mean Squared Error	0.2914
Overall TAS for the Dataset	0.650

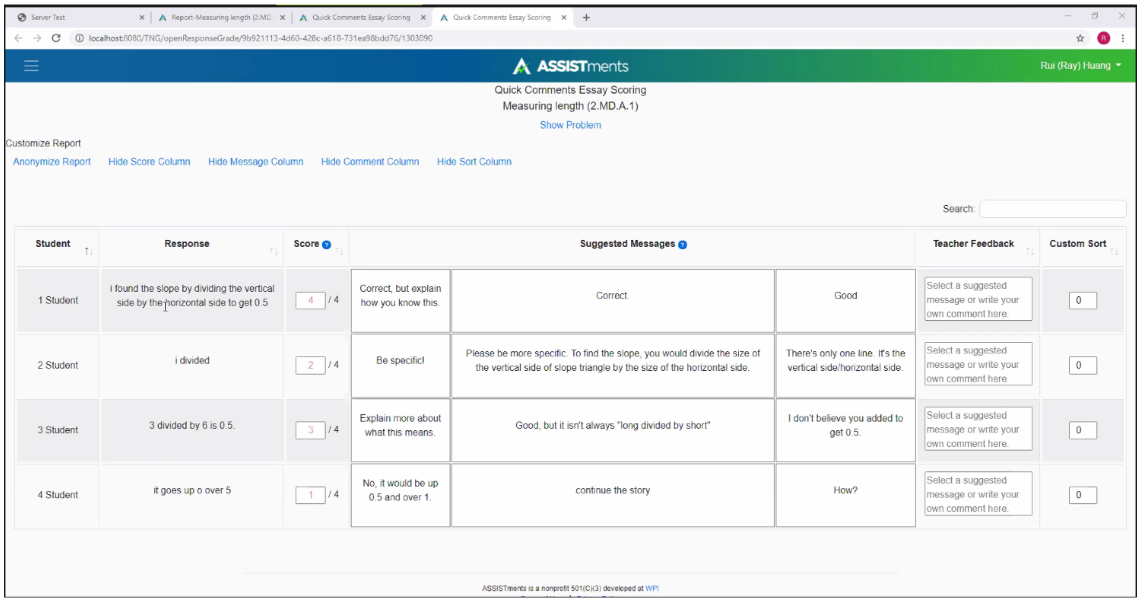
Table 3.4: Linear Regression Results

We achieved an overall TAS score of 0.65 which is closer to SBERT-Canberra results. And an RMSE of 0.2914. We were expecting this model to score better since this was supposed to be an ensemble of all the best approaches. Perhaps there was a correlation between SBERT-Canberra distance scores and TAS scores. Or may be Normalizing the distances would have helped. We can definitely investigate further into this as future work. We opted to implement the SBERT-Canberra Model in the ASSISTments infrastructure considering the training times and implementation ease of both the models.

3.7 BERT-Canberra Model in ASSISTments

3.7.1 Current Implementation

The current Quick-Comments feature in ASSISTments is updated with BERT-Canberra Model implementation. The model provides 3 suggestions from previously seen feedback comments of the same problem set. To ensure the quality of suggested comments, we set an upper threshold for Canberra distance when comparing answers beyond which the answer's comments are considered to be of lower confidence. To calculate the threshold, we compute the mean of distances which are sorted in ascending order and mark 1.5 std deviation apart from the mean. Set the threshold to be the lower 1.5 std deviation mark. We consider all the answers only below this threshold as potential candidates for comment suggestion. The closest answer's comments within the threshold are used as suggestions. If not enough answers within the threshold, the hard-coded comments are used to fill the gap.


Server Test
Report-Measuring length (2.MD...
Quick Comments Essay Scoring
Quick Comments Essay Scoring
localhost:8080/TNG/openResponseGrade/9b921113-4d60-428c-a618-731ea98bdd76/1303090
ASSISTments
Rui (Ray) Huang

Quick Comments Essay Scoring
Measuring length (2.MD.A.1)
Show Problem

Customize Report
Anonymize Report Hide Score Column Hide Message Column Hide Comment Column Hide Sort Column

Search:

Student	Response	Score	Suggested Messages			Teacher Feedback	Custom Sort
1 Student	i found the slope by dividing the vertical side by the horizontal side to get 0.5	4 / 4	Correct, but explain how you know this.	Correct.	Good	Select a suggested message or write your own comment here.	0
2 Student	i divided	2 / 4	Be specific!	Please be more specific. To find the slope, you would divide the size of the vertical side of slope triangle by the size of the horizontal side.	There's only one line. It's the vertical side/horizontal side.	Select a suggested message or write your own comment here.	0
3 Student	3 divided by 6 is 0.5.	3 / 4	Explain more about what this means.	Good, but it isn't always "long divided by short"	I don't believe you added to get 0.5.	Select a suggested message or write your own comment here.	0
4 Student	it goes up o over 5	1 / 4	No, it would be up 0.5 and over 1.	continue the story	How?	Select a suggested message or write your own comment here.	0

ASSISTments is a nonprofit 501(C)(3) developed at WPI

Figure 3.6: ASSISTments Quick Comments with Sentence-BERT-Canberra

3.7.2 Pilot Observational Data Analysis

We enabled the updated feature for certain duration for 8 teachers and collected a preliminary set of teacher usage data. This dataset contains 20,053 samples of teacher and student data along with student answer, teacher comment and suggested comments. A total of 631 students data was collected from ASSISTments database.

Comparison Method	**Mean score**
No Edits	0.112
As is or with Edits	0.126
Longest Sub-string Match	0.242
Normalized Compression Distance	0.185

Table 3.5: Encoder-Distance Comparison Reports with max possible score

The system supports opting to edit the suggested answer by teachers. The teacher is free to either choose a suggested comment with or without editing it or write entirely their own feedback. To get a sense of how the feature is being used, we analyzed the dataset to match each teacher provided comment to predicted comments using the implemented BERT-Canberra model. Using the problem ID and student answer, we predicted the suggested comments for each of the student answer using our previous dataset. While there are many ways to compare the match between two strings/sentences, we have employed the methods shown in table 3.5 to learn about the percent of times the suggestions being used.

First, we check for the number of times one of the predicted suggestions being used as is i.e, without any edits. We simply loop through all the predicted comments and see if one of them completely matches with the teacher provided comment. If it matched, we scored it 1 otherwise 0. Taking a mean of all the scores resulted in the mean score of 0.112 indicating that at least 11% of the times the suggested comments are used as is.

To accommodate for edits made by the teacher, we check if one of the predicted comments is a continuous sub-string in the teacher comment accounting for edits made at the beginning or at the end. After scoring, we saw a mean score of 0.126, which isn't a lot of improvement compared to 'No Edit' comparison method.

Next we look for longest matching sub-string to account for cases where a suggested comment is edited to remove/replace a part of it. We compute the longest sub-string match and consider only the comments that contain more than 50% of the predicted comment to be scored as 1. The mean score resulted to be 0.24. This indicates about at least 24% of the times the suggested comment is being used as a reference or base comment structure.

Another comparison method we use is the Normalized Compression Distance (NCD)[14] method. NCD is a way of measuring similarity between two objects represented as file objects in bit representation. It is measured as the ease with which one object can be transformed into another. NCD is being used in several applications like plagiarism checker, which motivates us to use it in our analysis. While scoring, we only consider the comments which require less than 50% effort to transform the object into another. The resulting mean score of NCD was 0.185.

These methods are used to just get a fair idea of the model usage in the real-time scenario so we could improve upon the model and keep increasing the usage percentage. With that thought, we proceeded to implement an infrastructure that caters to all researchers needs to run an experiment in Quick-Comments to further improve the model accuracy of the overall feature in ASSISTments.

Chapter 4

RCT Logging Infrastructure

4.1 Randomized Control Trials

Randomized Control Trials (RCT) are a way of experimenting with the newly built systems to study their efficacy. The users of the system are subjected to either a controlled system or a treated system. The controlled system is where the users continue to use the traditional version of the system without any new features, whereas the treated system is the one where the users are subjected to a specific user condition.

4.2 QUICK-Comments RCT

The implementation of this RCT logging infrastructure is aimed to create a single data storage place for multiple studies conducted using Quick-Comments. We have designed the database in such a way that makes it generic to fit the needs of all logging experiments. The teachers log into their ASSISTments accounts and grade their students' submissions as they usually do on ASSISTments. The system logs all actions, namely the student answer, the answer grade, suggested responses, the

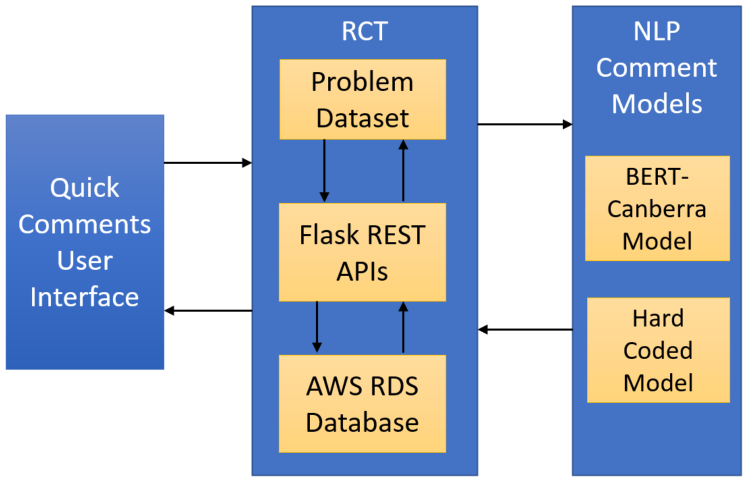


Figure 4.1: Quick Comments RCT High-Level Design

user condition, the submitted feedback comment.

The teacher actions are communicated using REST APIs implemented with Python Flask. When the teacher launches the grading page, the API **"qc_generate_comment"** loads the student data containing teacher id, problem id, student ids, student answers and returns with a list of suggested comments for each of the student answers.

qc_generate_comment(problems, response_list, request) where,

problems: List of all problem IDs found in loaded json file.

response_list: List of dictionaries, with number of dictionaries matching number of responses in data.

request: Deserialized json file containing student response data.

return: List of dictionaries where each dictionary is a student response and relevant related data.

The RCT is embedded within the Flask Application and supports user condition randomizations at teacher, student, assignment, or mixed level. With this layer researchers will have the ability to randomize which models are used based on the given user conditions. Each study can have multiple user conditions and each user condition can be associated with multiple models. Each study has predetermined associations with the randomization attributes. When a request comes in, first we check if a condition has already been assigned to the given attributes. If condition exists, we read the conditions and return the appropriate model for comment generation. If condition does not exist, we randomly pick a user condition at run-time to ensure equal likelihood based on assignment probability set for the study. Log the user condition, and return the user condition model for comment generation.

4.2.1 Database Design

In figure 4.2, the database ER diagram describes the relationship between the tables. The upper half tables colored dark blue are used to log student data, teacher data, suggested comment data, model data and request metadata. The lower half tables in light blue are used to store the user conditions, randomizations, study data and study status. Below we provide description of the main tables.

- **Request_logs**: Stores the request metadata when a POST request is made from the client side.
- **Request_problem_logs**: Stores the teacher id and problem ids associated with the request.
- **Student_answers**: Stores the student answers, student id, associated grade, number of comments suggested for each problem id.
- **Comments**: Stores suggested comments, comment order, teacher comment for each student answer.
- **Models**: Stores all the models developed for Quick-Comments and their descriptions.

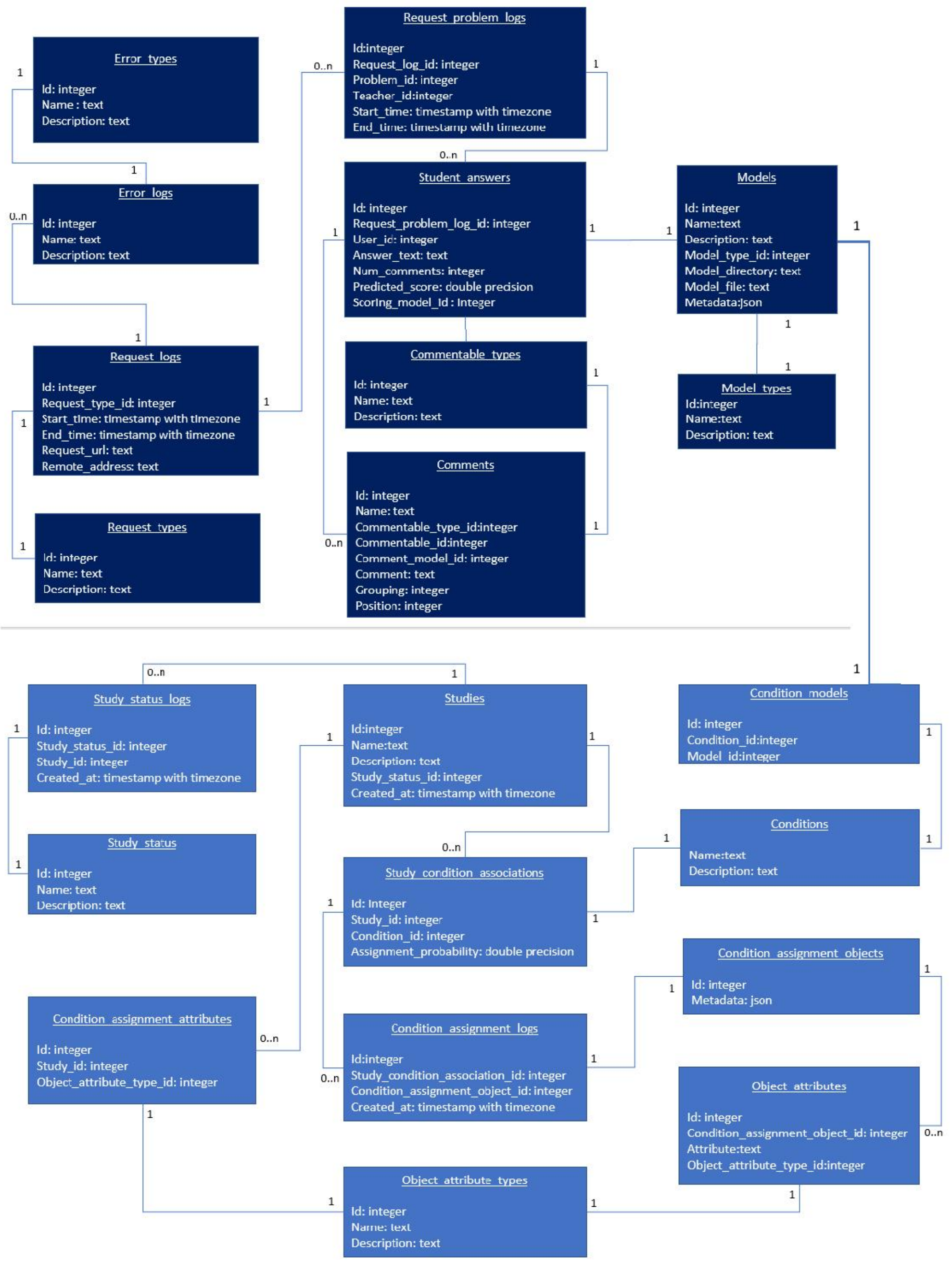


Figure 4.2: ER Diagram of RCT

- **Studies**: Stores each RCT logged in as an individual study.
- **Study_status_logs**: Stores the status of each study. The status can be in one of the following. 1. Started 2. Running 3. Halted 4. Finished.
- **Conditions**: Stores all the conditions a study is subjected to.
- **Condition_models**: Stores the condition and model associations.
- **Study_condition_associations**: A study can have more than 1 condition. This table stores all the user conditions associated with the study.
- **Condition_assignment_objects**: Stores an object i.e a record to map to all associated object attributes predetermined for a study condition.
- **Object_attributes**: Stores the attribute value for each object attribute type.
- **Object_attribute_types**: Stores the types of object attributes. Currently the RCT uses 1. Teacher type 2. Student type 3. Assignment type.
- **Condition_assignment_logs**: Stores the study-condition associations to condition assignment objects.
- **Condition_assignment_attributes**: Stores all the attribute types the study is associated with.

4.2.2 Randomizing User Conditions

RCT supports setting the number of suggested comments as required by the study. The conditions required by the study must be logged before running the study and the table study_condition_associations populated accordingly. The study_condition _associations table also requires the researchers to set the assignment probability for each associated condition which is the weight assigned to the condition when randomly selecting a user condition for a new request. Researchers must agree beforehand on which attributes they intend to randomize the study and accordingly associate the object attribute types with the study. The object attributes could be teacher id, student id, problem id or a mix of these. Suppose we are randomizing a study on student id and problem id, we could create a condition_assignment_object and use the condition_assignment_object_id to link to two object_attribute records. One for storing student id as attribute value which is of type 'student', and the other for storing problem id as attribute value which is of type 'problem'. This condition_assignment_object is associated with the chosen user condition and logged into condition_assignment_logs. Figure 4.3 depicts a high level Sequence diagram of Randomizing attributes in the RCT logging infrastructure.

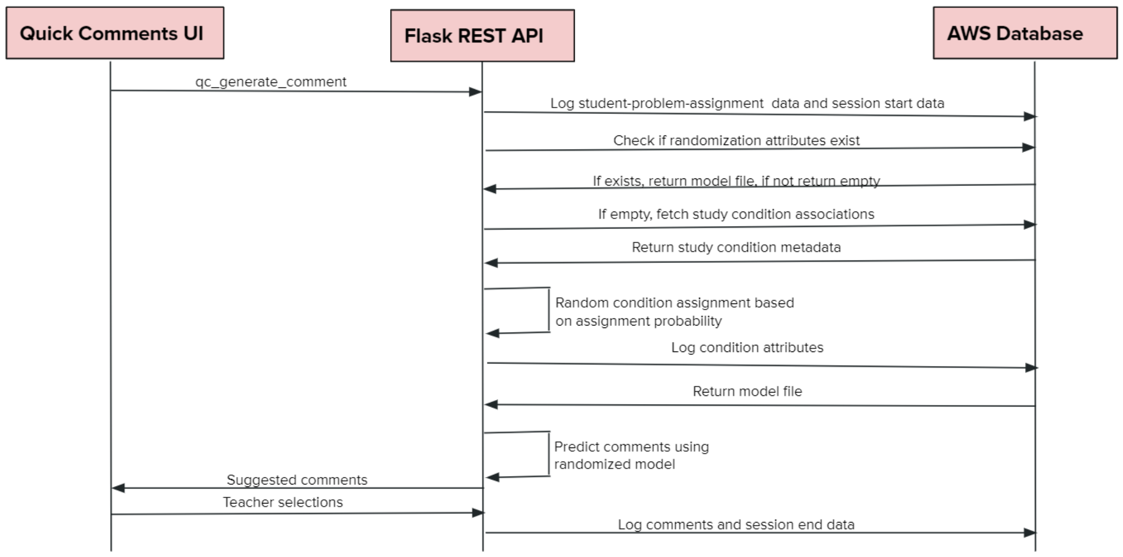
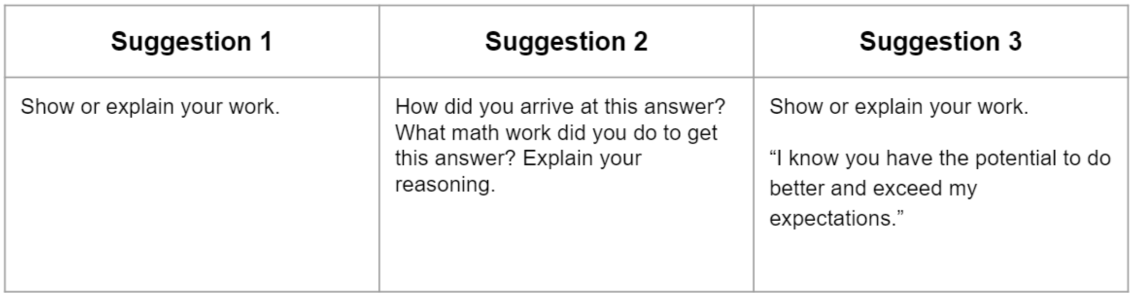

Quick Comments UI
Flask REST API
AWS Database
qc_generate_comment
Log student-problem-assignment data and session start data
Check if randomization attributes exist
If exists, return model file, if not return empty
If empty, fetch study condition associations
Return study condition metadata
Random condition assignment based on assignment probability
Log condition attributes
Return model file
Predict comments using randomized model
Suggested comments
Teacher selections
Log comments and session end data

Figure 4.3: Randomization Sequence of Events

4.3 BERT-Canberra-Wise Model

Wise feedback refers to psychological interventions that convey a genuine belief of teachers' high expectations from the students. A study conducted on the effects of Wise feedback[14] in which 7th-grade students received critical feedback from their teacher that, in the treatment condition, was designed to lessen mistrust by emphasizing the teacher's high standards and belief that the student could meet those standards, a strategy known as wise feedback. Wise feedback increased students' likelihood of submitting a revision of an essay and improved the quality of their final drafts.

Our implementation simply uses a list of predefined Wise preambles that at random, one of them will be appended to the suggested response. The user condition here would be to subject teachers to responses with and without the Wise Preamble appended to the suggested response. The system logs the actions of the teacher in a database of the infrastructure. The running of the WISE RCT is left as a scope for future work.



Suggestion 1	Suggestion 2	Suggestion 3
Show or explain your work.	How did you arrive at this answer? What math work did you do to get this answer? Explain your reasoning.	Show or explain your work. "I know you have the potential to do better and exceed my expectations."

Figure 4.4: Example of Wise suggestions

Chapter 5

Limitations and Future Work

We have developed a model that recommends responses which are trained at a problem level, this means the model recommendation is limited to only previously seen problems. The problem-wise recommendation approach seems promising when compared to across-problem recommendation. If a new problem comes through for which there is no previously recorded data, the model is unable to suggest any feedback comments. We intend to keep adding more answers to each problem set and new problems to the dataset for future experiments. The model is designed to do well for ASSISTments dataset structure. The same model might not score well on differently structured dataset. As a future work, we will also explore more methodologies to provide suggestions for unseen problems and explore ways to generalise models for unstructured data. We will also explore an RCT to study the implications of WISE feedback on student learning by utilizing the Wise model.

Chapter 6

Conclusion

In this research we have demonstrated a model for recommending responses to open-ended student answers. We have analysed several encoders and distance metrics that work well for the dataset at hand. We identified a methodology to evaluate the recommending model and found SBERT-Canberra model to be the best performing model with 65% teacher agreement. Through the teacher usage data analysis we roughly estimated that a small percent about 11% to 24% of suggested comments were indeed useful in reducing teachers grading effort. To improve the efficiency of the model we have developed an RCT infrastructure that allows researchers to randomize models and user conditions as required by the study.

Chapter 7

References

[1] Ahmad, Kashif Qadir, Junaid Al-Fuqaha, Ala Iqbal, Waleed Elhassan, Ammar Benhaddou, D. Ayyash, Moussa. (2020). Artificial Intelligence in Education: A Panoramic Review. 10.35542/osf.io/zvu2n.

[2] Roschelle, Jeremy, Mingyu Feng, Robert F. Murphy, and Craig A. Mason. 2016. "Online Mathematics Homework Increases Student Achievement." AERA Open 2 (4): 2332858416673968.

[3] Foltz, Peter Laham, Darrell Landauer, T.. (1999). Automated Essay Scoring: Applications to Educational Technology. World Conference on Educational Multimedia, Hypermedia and Telecommunications. 1.

[4] Leacock, C., Chodorow, M. C-rater: Automated Scoring of Short-Answer Questions. Computers and the Humanities 37, 389–405 (2003).

https://doi.org/10.1023/A:1025779619903

[5] Chakraborty, Udit Gurung, Rashmi Roy, Samir. (2014). Semantic Similarity Based Approach for Automatic Evaluation of Free Text Answers Using Link Grammar. Proceedings - IEEE 6th International Conference on Technology for Education, T4E 2014. 10.1109/T4E.2014.57.

[6] Kumar, Sachin et al. “Earth Mover’s Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading.” IJCAI (2017).

[7] Kannan, Anjuli Young, Peter Ramavajjala, Vivek Kurach, Karol Ravi, Sujith Kaufmann, Tobias Tomkins, Andrew Miklos, Balint Corrado, G.s Lukacs, Laszlo Ganea, Marina. (2016). Smart Reply: Automated Response Suggestion for Email. 955-964. 10.1145/2939672.2939801.

[8] Reimers, Nils, and Iryna Gurevych. ”Sentence-bert: Sentence embeddings using siamese bert-networks.” arXiv preprint arXiv:1908.10084 (2019).

[9] Cer, Daniel Matthew et al. “Universal Sentence Encoder.” ArXiv abs/1803.11175 (2018): n. pag.

[10] Devlin, J. et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” NAACL-HLT (2019).

[11] Babar, N. (2018, October 2). The Levenshtein Distance Algorithm - DZone Big Data. https://dzone.com/articles/the-levenshtein-algorithm-1

[12] Giuseppe Jurman; Samantha Riccadonna; Roberto Visintainer; Cesare Furlanello; ”Canberra Distance on Ranked Lists”, in Shivani Agrawal; Chris Burges; Koby Crammer (editors); Proceedings, Advances in Ranking – NIPS 09 Workshop, 2009, p. 22–27

[13] Walton, G. M. (2014). The new science of wise psychological interventions. Current Directions in Psychological Science, 23(1), 73–82.

https://doi.org/10.1177/0963721413512856

[14] Yeager, D. S., Purdie-Vaughns, V., Garcia, J., Apfel, N., Brzustoski, P., Master, A., . . . Cohen, G. L. (2014). Breaking the cycle of mistrust: Wise interventions to provide critical feedback across the racial divide. Journal of Experimental Psychology: General, 143(2), 804-824. http://dx.doi.org/10.1037/a0033906