Using ASSISTments to Help Teachers Recover COVID19-Related Learning Loss

An Interactive Qualifying Project submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of Bachelor of Science

By: Shundong Li, Gong Fan

October 13, 2021

Report Submitted To:

Professor Neil Heffernan, Worcester Polytechnic Institute

Abstract

The COVID pandemic hit heavily on all areas of industries, especially on education. Professor Neil Heffernan, also the founder of ASSISTments, enlisted our help to recoup the loss that occurred during the pandemic. ASSISTmentsis a tool for both teachers and students to use and improve their efficiency. Teachers can easily assign homework and analyze students' performance or learning experience, where students can have an easier time understanding the topic by getting prompt feedback from ASSISTments. Our team aims to make this process even more efficient on the teacher's side by providing a tool to predict the score of a student's answer and give suggestions for comments. We worked on the training of NLP models that would be deployed into classrooms closely after the end of this project. The NLP model aims to give better and more accurate scores and comments suggestions to teachers. The Bert transformer method is used in this NLP model. We also looked into methods of data parsing and other model ensembles to compare with the models we plan to deploy

Table of Contents

List of Figures

List of Tables

Introduction

Natural Language Processing (NLP) is a dedicated branch of machine learning, artificial intelligence in the direction of linguistics. The application of NLP ranges from processing speech and text like optical character recognition to high-level applications like Apple Siri and Amazon Alexa. The general goal for NLP is making the computer capable of "understanding" the natural language contents. Due to the vastly different goals of each NLP application, the definition of "understand" varies from task to task.

ASSISTments is a free public service aimed to serve schools and teachers with better maths teaching. Despite the marketing, ASSISTments also have data and research into other topics of tutoring. The quick comments service our team has been assigned aims to build an online platform that could take in previous students' answers, teachers' grading, and replies and predict the scores of new students who take the same question. Grading and commenting on the open answer questions is much more complicated than multiple choice answers because there is no "correct answer." Even if there are keywords for each question, the logistics between the keywords for each question varies tremendously, making regular parsing and checking wildly inaccurate. Not to mention there are usually multiple approaches for one math problem. Therefore, the modeling and prediction process relies on the NLP technology we have introduced previously.

Quick comments service is already in the production stage in ASSISTments to help predict open answer math problems. Professor Heffernan wants to deploy it to the CS2223 algorithms course at WPI. But deploying the models previously trained on math questions to algorithm quizzes makes no sense. Therefore, our team aims to train the models for the course and assess if the models are reliable enough to deploy.

Background

Package setup (venv & VScode)

Setting up the packages needed for the quick comments services can be problematic if we install them directly onto the python we currently use. Several packages rely on Python 3.7 that the previous developers worked on before, and specific packages we currently use also conflict with the requirements.

We have tried parallel installing Python 3.7 onto our machine, and it still ended up with weird conflicts when we proceeded to install the packages. The most frustration-free approach was to use virtualenv to create a virtual Python environment to install the packages. The steps are shown on the next page. However, setting up with the steps below still requires the developers to activate and deactivate each time, and we came up with a solution that resolves this hassle.

# Create virtualenv (One-time setup)

- *virtualenv -p python3 venv*
- (if default python3 version is not python 3.7 and parallels installed another python 3.7 instances, change the command with *virtualenv -p python3.7 venv*)
- *Add venv in .gitignore file*

# Activate virtualenv

- *source venv/bin/activate*

# Install from requirements.txt

- *pip install -r requirements.txt*

# Generate requirements.txt after each new installation before pushing to git

- *pip freeze > requirements.txt*

# Deactivate

- *deactivate*

Since both of us on the team use VScode to develop, we only looked into methods that work with VScode. After installing the python plugin for VScode from Microsoft, we can set the default python interpreter for each project after opening a python file by clicking the python version at the lower left of the screen. Or we can use the shortcut ctrl+shift+P to open the command palette and type in *python: Select Interpreter.*

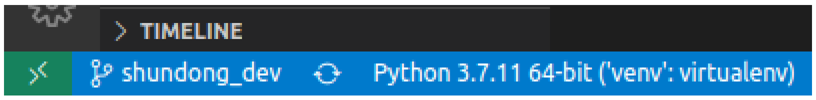
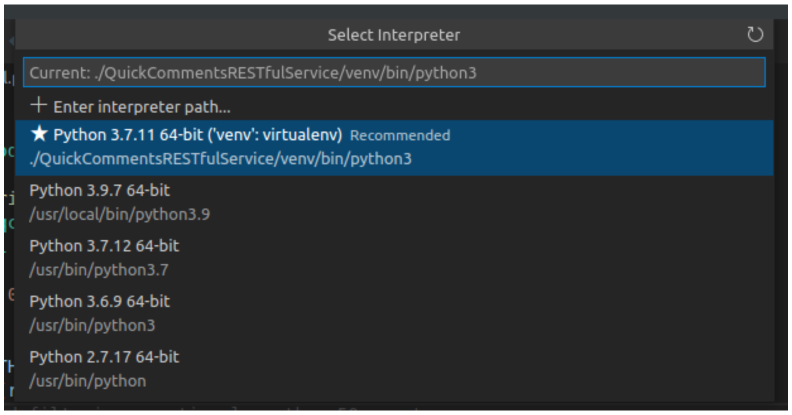


Figure 1, an example of using the venv python 3.7.11 as an interpreter

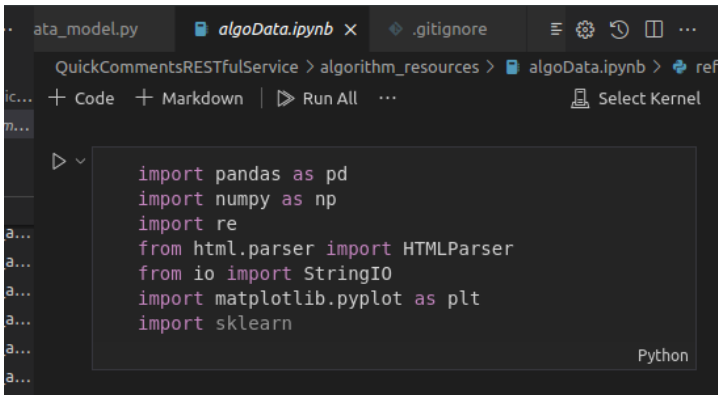
Both methods will open the select interpreter pop-up shown below. We can then navigate to the path where we installed the virtual environment and enter it. After doing this, every time we hit run in VScode, it will run with the designated virtual environment.



```
Select Interpreter
Current: ./QuickCommentsRESTfulService/venv/bin/python3
+ Enter interpreter path...
★ Python 3.7.11 64-bit ('venv': virtualenv) Recommended
./QuickCommentsRESTfulService/venv/bin/python3
Python 3.9.7 64-bit
/usr/local/bin/python3.9
Python 3.7.12 64-bit
/usr/bin/python3.7
Python 3.6.9 64-bit
/usr/bin/python3
Python 2.7.17 64-bit
/usr/bin/python
```

Figure 2, the example of select interpreter path page

Also, suppose we want to use Jupyter Notebook to develop. In that case, the workaround is similar: we first install the Jupyter Notebook Renderers plugin from Microsoft, we then navigate to an ipynb file, and on the upper right corner, there is a button for us to select the python kernel for the notebook. The remaining setup is similar to the previous setup with the



```
import pandas as pd
import numpy as np
import re
from html.parser import HTMLParser
from io import StringIO
import matplotlib.pyplot as plt
import sklearn
```

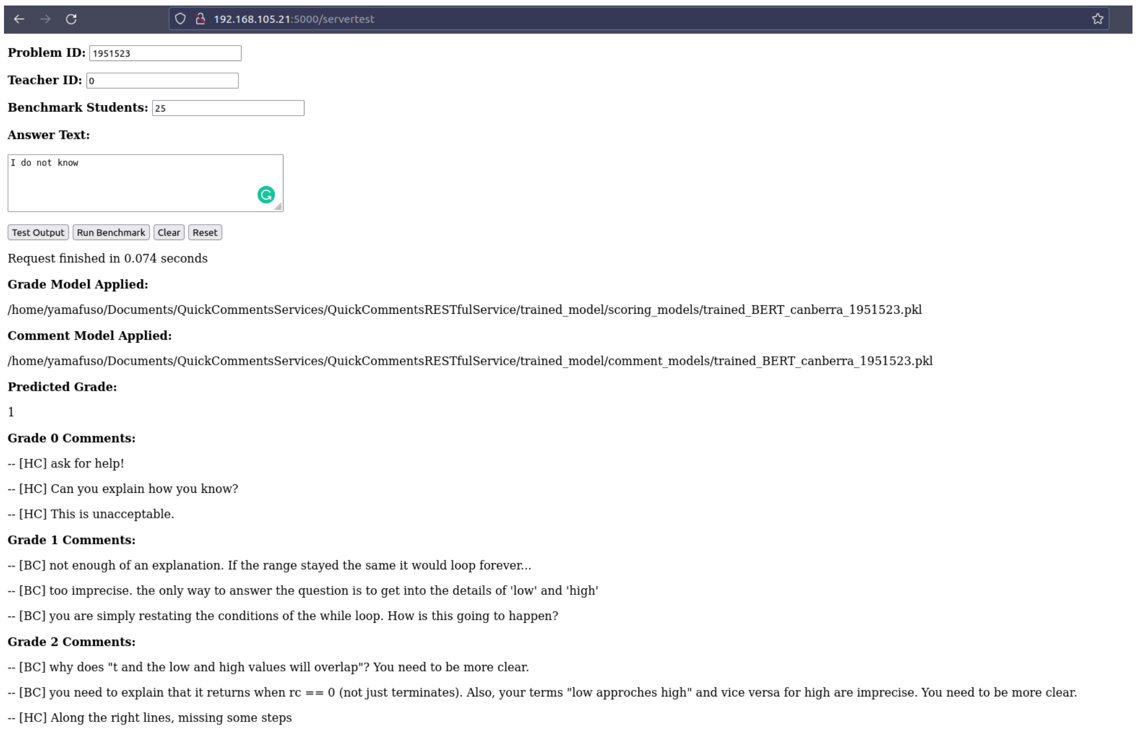
python interpreter.

Figure 3, the example of the select kernel for jupyter notebook

Setting up the environment is crucial before developing, especially in a short timeframe like seven-week IQP; we hope our report can also help future students who continue our work.

Model Prediction Example (Localhost)

Currently, the quick comments service is developed with flask micro-framework. To run the app, we can execute *flask run* or execute the *app.py* file. The default port for the localhost is 5000. We will only see the default page with "Server is running" if we go to *localhost:5000.* If we go to *localhost:5000/servertest*, we can then access the test page for the service. We have also used this page to validate our models are working as expected by plugging in our unique results and seeing how the model predicts. Below is an example of the servertest page:



Problem ID: 1951523

Teacher ID: 0

Benchmark Students: 25

Answer Text:

```
I do not know
```

Request finished in 0.074 seconds

Grade Model Applied:

/home/yamafuso/Documents/QuickCommentsServices/QuickCommentsRESTfulService/trained_model/scoring_models/trained_BERT_canberra_1951523.pkl

Comment Model Applied:

/home/yamafuso/Documents/QuickCommentsServices/QuickCommentsRESTfulService/trained_model/comment_models/trained_BERT_canberra_1951523.pkl

Predicted Grade:

1

Grade 0 Comments:

-- [HC] ask for help!

-- [HC] Can you explain how you know?

-- [HC] This is unacceptable.

Grade 1 Comments:

-- [BC] not enough of an explanation. If the range stayed the same it would loop forever...

-- [BC] too imprecise. the only way to answer the question is to get into the details of 'low' and 'high'

-- [BC] you are simply restating the conditions of the while loop. How is this going to happen?

Grade 2 Comments:

-- [BC] why does "t and the low and high values will overlap"? You need to be more clear.

-- [BC] you need to explain that it returns when rc == 0 (not just terminates). Also, your terms "low approches high" and vice versa for high are imprecise. You need to be more clear.

-- [HC] Along the right lines, missing some steps

Figure 4, an example of the servertest page

BERT transformer models

BERT stands for Bidirectional Encoder Representations from Transformers. The transformer means a deep learning model that each output node feeds back to the input nodes. The weights for each node also change with each evaluation. On the other hand, bi-directional means that the model can read the text input beginning at the right or left. This feature is made possible because of the transformer.

Logistics regression

Logistics regression is a statistical model that utilizes a function that takes in data and generates discrete regression results. This modeling fits perfectly for the quick comments services because instead of continuous results, we would like our prediction of scores to be integers within [0, 4]. At least for the comment service, we can create simple logistics regression to compare with the BERT models.

Regex

Regex stands for the regular expression. A regular expression is a sequence of characters that defines a search pattern. A program can use an expression to look for matching groups inside a string and isolate them. Regex can be helpful in input validations or looking for specific patterns to be replaced or removed.

Ten-fold cross-validation

The cross-validation is an approach to divide our dataset into equal size “folds” randomly, and for each fold, we train the model with the rest of the “folds” and predict the result onto the specific fold. We use ten-fold cross-validation in our case. The validation process is like this:

- Randomly assign numbers within [1, 10] for each row in the dataset; these numbers represent each row’s fold.
- Starting with fold 1, we train the model in fold 2 to 10 and predict the answers for fold 1
- Move to the next fold, train again with the reset, predict. Loop until all folds are predicted.
- Save all the predicted results back to the original file.
- Run the testing functions

Professor Heffernan stressed multiple times that as time moves forward, there will always be newer models and methods to process and predict the dataset, like how machine learning evolved to deep learning; simple transformer models changed to BERT transformer models. But cross-validation would always be applicable to verify if the model holds water. And when processing the data, it is crucial to get some good results, and it is more important to make sense of the generated good result.

Methodology

1. Separate code form answers

Currently, the QuickComment Service uses the Bert NLP model to predict comments and scores for student answers. However, the problem with this model is that code is treated the same as other natural languages; this could be a point of improvement to increase the accuracy of the prediction of comments and scores.

To separate the code from text, regex expressions are used to identify code from student answers. Then the built-in function of the regex library can separate the identified portion into another column in the data.

Code has some pacific patterns, such as parentheses or other special characters and patterns that normally do not appear in texts. Regex expressions can capture those groups such as (), =>, if('condition'), etc.

2. Model Training

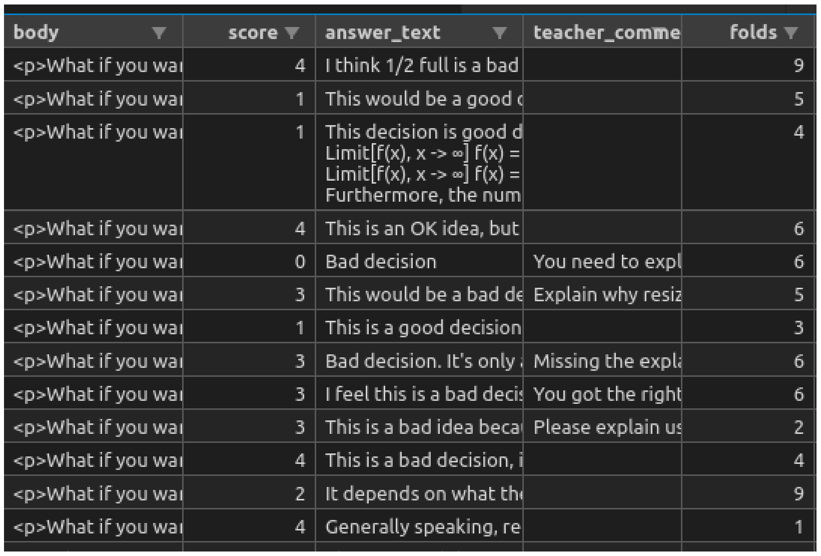
The other difficulty when training models on the algorithm course data is the data parsing. Previous quick comments services fetch the required data from the ASSISTments SQL database directly, but we received the data directly in CSV format. Since the Python class for both models also fetches data from the SQL directly, we decided to create two new classes for the algorithm models. We saved the new models into the script "SBERT_canberra_algorithm_model.py"

After we settled with the model classes, we developed five functions for the different process stages: data cleaning, code parsing, data training, data training for the ten-fold, and finally, model testing. Detailed explanations are shown below:

- data cleaning

We first read the CSV file and store it into a data frame object *data_df*; we then take out the problem id column to get all unique ids. To ensure the reliability of the model, we would drop any model with less than 50 student responses. We then assign random fold tags (from 0 to 9) to a new column named *folds* for later ten folds testing. We also strip the HTML tags from students' comments because the response is fetched from ASSISTments directly. Finally, we would save *data_df* into a new CSV file with the original filename attached with *"_randomized"* in the end.

This function also returns the *data_df* object for potential references. When detecting there is already a column named *folds* in the data frame, we would say that this dataset is already cleaned and return the *data_df* directly without any of the processing listed above.

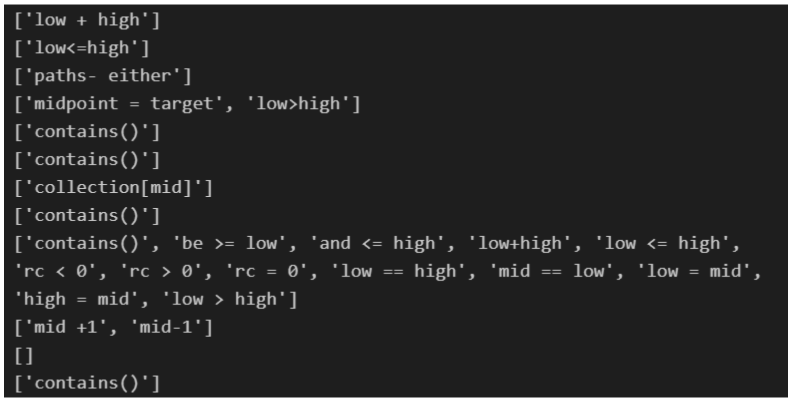


body	score	answer_text	teacher_comme	folds
<p>What if you wa	4	I think 1/2 full is a bad		9
<p>What if you wa	1	This would be a good c		5
<p>What if you wa	1	This decision is good d Limit[f(x), x -> ∞] f(x) = Limit[f(x), x -> ∞] f(x) = Furthermore, the num		4
<p>What if you wa	4	This is an OK idea, but		6
<p>What if you wa	0	Bad decision	You need to expl	6
<p>What if you wa	3	This would be a bad de	Explain why resiz	5
<p>What if you wa	1	This is a good decision		3
<p>What if you wa	3	Bad decision. It's only	Missing the expl	6
<p>What if you wa	3	I feel this is a bad deci	You got the right	6
<p>What if you wa	3	This is a bad idea beca	Please explain us	2
<p>What if you wa	4	This is a bad decision, i		4
<p>What if you wa	2	It depends on what th		9
<p>What if you wa	4	Generally speaking, re		1

Figure 5, an example of a cleaned dataset (note the uncleaned HTML tag in the body column)

- code parsing

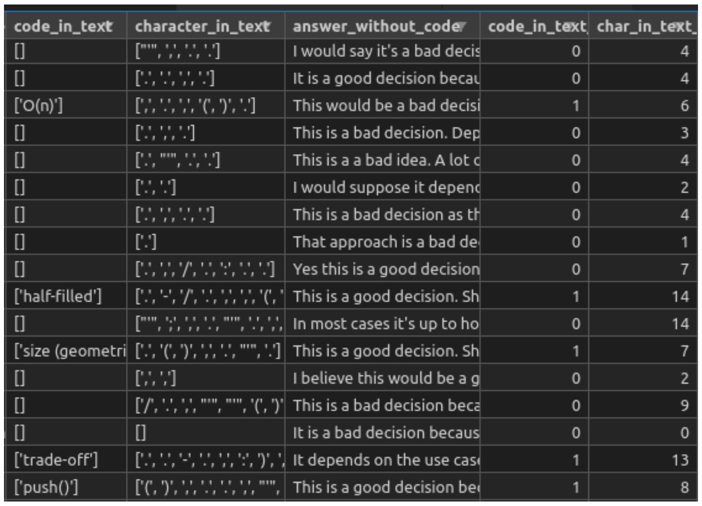
This function is our implementation of using regex to parse out the code from responses. As described in methodology subsection one, code has been isolated from student answers using regex expressions.



```
['low + high']
['low<=high']
['paths- either']
['midpoint = target', 'low>high']
['contains()']
['contains()']
['collection[mid]']
['contains()']
['contains()', 'be >= low', 'and <= high', 'low+high', 'low <= high',
'rc < 0', 'rc > 0', 'rc = 0', 'low == high', 'mid == low', 'low = mid',
'high = mid', 'low > high']
['mid +1', 'mid-1']
[]
['contains()']
```

Figure 6, an example of parsed out code from the answer

For comparison, we also saved a column of answers without the code to make a comparison model with the original one. We also counted the number of codes and characters in the answer and saved them for the simple logistics regression.



code_in_text	character_in_text	answer_without_code	code_in_text	char_in_text_
[]	["'", ',', '.', '.']	I would say it's a bad decis	0	4
[]	['.', '.', ',', '.']	It is a good decision becau	0	4
['O(n)']	[',', '.', ',', '(', ')', '.']	This would be a bad decisi	1	6
[]	['.', ',', '.']	This is a bad decision. Dep	0	3
[]	['.', "'", '.', '.']	This is a a bad idea. A lot c	0	4
[]	['.', '.']	I would suppose it depenc	0	2
[]	['.', ',', '.', '.']	This is a bad decision as th	0	4
[]	['.']	That approach is a bad de	0	1
[]	['.', ',', '/', '.', ':', '.', '.']	Yes this is a good decision	0	7
['half-filled']	['.', '-', '/', '.', ',', ',', '(', '	This is a good decision. Sh	1	14
[]	["'", ';', ',', '.', "'", '.', ',',	In most cases it's up to ho	0	14
['size (geometri	['.', '(', ')', ',', '.', "'", '.']	This is a good decision. Sh	1	7
[]	[',', ',']	I believe this would be a g	0	2
[]	['/', '.', ',', "'", "'", '(', ')'	This is a bad decision beca	0	9
[]	[]	It is a bad decision becaus	0	0
['trade-off']	['.', '.', '-', '.', ',', ':', ')', ',	It depends on the use case	1	13
['push()']	['(', ')', ',', '.', '.', ',', "'",	This is a good decision bec	1	8

Figure 7, an example of added feature columns

- data training & training for the ten-fold

We had two functions to train the data; the first one does not use ten-fold cross-validation. Therefore all data for each question id will be used and trained; these models are mainly used to deploy back to ASSISTments. But like we said before, if we do not perform any kind of validation, we cannot make sure the models mainly designed for math problems can handle computer science algorithm questions with code well. Therefore, we have another function to train the ten-fold models.

Both functions will detect if there are already any existing models for the specific question id in the designated path. If that is the case, the direct training function will skip to the next id, while the ten-fold function will read in the model, perform prediction on the corresponding fold, and finally update the file with the predicted answers. If there is no such model, the production training function will train the model for that id, and the ten-fold function will train the *ten* models for that id.

- model testing

Since we already have the student received answers from the dataset and the predicted answer from the model generated by the process shown above, we can run the tests for AUC, RMSE, and cohen_kappa.

Data Analysis

1. Grading Models

	AUC	RMSE
With code	0.719	1.229
Without code	0.703	1.242

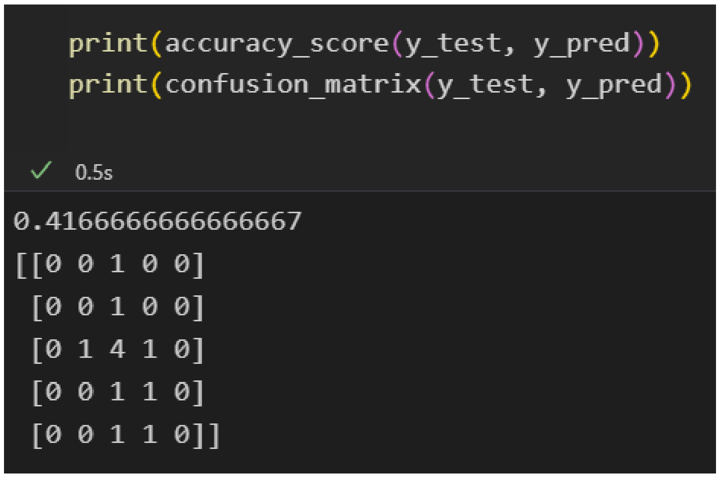
Table 1, validation result for the models trained differently

For the table above, the first column with/ without code means that whether the student_answer column we trained with has the code part purged or not. The second column, AUC, stands for the AUC score, which measures the total area under the ROC curve. In other words, for example, we have a 71.867% chance to correctly predict the student score from the ten-fold cross-validation with the model "with code." The third column, RMSE, stands for the root mean square deviation, indicating how our data fits our model. The RMSE of 1.22 or 1.24 means that the model would generally predict the answer around 1.2 points off the actual score.

The table on the previous page shows our model performance did not change greatly even though we purged the student answer code. This might be partly because the amount of code is just a mere fraction compared to the total answer length. Also, the answering logic is mainly conveyed by the language instead of the code, especially within restricted length problems. But this also gives us the idea of separating the code and ensemble the non-code SBERT model with logic regression or other models about the code. We did not fully complete the process, and the future team could start from there.

2. Code Feature Logistic Regression

After the student response has been parsed to code and text, logistic regression to predict an answers' score is performed on feature columns of the number of code segments and the number of special characters in the text. The data set is slipped into ten folds, with one fold for testing. The logistic regression is performed with the Scikit-learn algorithm, and we will obtain a classifier. Below is the accuracy of our classifier and the confusion matrix generated.



```
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

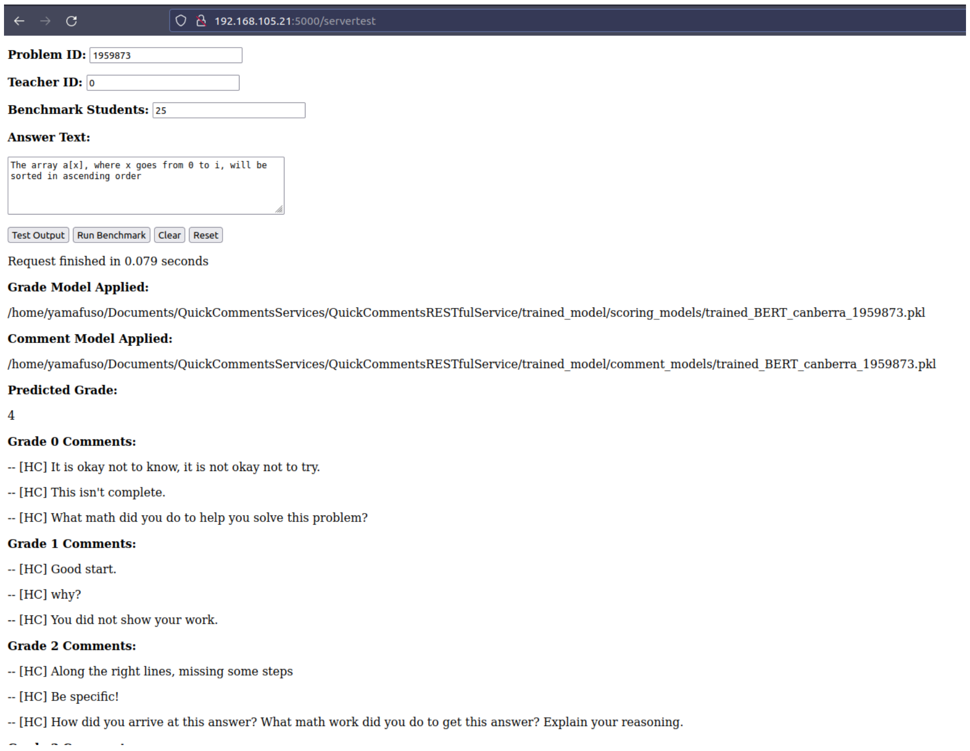
```
✓ 0.5s
0.4166666666666667
[[0 0 1 0 0]
 [0 0 1 0 0]
 [0 1 4 1 0]
 [0 0 1 1 0]
 [0 0 1 1 0]]
```

Figure 8, Accuracy and confusion matrix for the logistic regression

As we can see, the accuracy is around 0.41, which is not so great. However, it is better than randomly guessing the score from 0-4, which is 0.2.

3. Manual Input

After the above analysis, we attempted to write our answers to see how the models perform from new samples. We altered the expressions from the given answers to ensure the given score did not deviate much from the actual score. Although we only made ten attempts over three questions, the model predicted perfectly from the questions we altered. Again, ten samples among three questions are nowhere statistically significant, but we are still quite satisfied with the result of the model.



Problem ID: 1959873

Teacher ID: 0

Benchmark Students: 25

Answer Text:

```
The array a[x], where x goes from 0 to i, will be
sorted in ascending order
```

Request finished in 0.079 seconds

Grade Model Applied:

/home/yamafuso/Documents/QuickCommentsServices/QuickCommentsRESTfulService/trained_model/scoring_models/trained_BERT_canberra_1959873.pkl

Comment Model Applied:

/home/yamafuso/Documents/QuickCommentsServices/QuickCommentsRESTfulService/trained_model/comment_models/trained_BERT_canberra_1959873.pkl

Predicted Grade:

4

Grade 0 Comments:

-- [HC] It is okay not to know, it is not okay not to try.

-- [HC] This isn't complete.

-- [HC] What math did you do to help you solve this problem?

Grade 1 Comments:

-- [HC] Good start.

-- [HC] why?

-- [HC] You did not show your work.

Grade 2 Comments:

-- [HC] Along the right lines, missing some steps

-- [HC] Be specific!

-- [HC] How did you arrive at this answer? What math work did you do to get this answer? Explain your reasoning.

Figure 9, one example of our altered input and predicted grade

Conclusions and Recommendations

1. Grading Models

Throughout this project, we first familiarized ourselves with NLP and the project. We then trained both commenting and grading models for the dataset. We worked towards the first evaluation steps for the comment models as we graded the answers and saw how the grading models assisted in improving these models. The next step would be working on the improvement of these models. For the grading models, we evaluated their performance with two conditions and designed simple logistic regression to compare them. We did not directly modify the models to yield better results; instead, we attempted to derivative code-related features from the students' answers. Hopefully, our work and this report will help the following team continue from our topic.

2. Future Work

Our team will also help the TAs for the incoming 2021 B-term algorithm course to use ASSISTments in the class.

Although we did not have the time to develop a robust way to ensemble code features into the existing SBERT models, future teams can start from here and see if the approach could give more accuracy. Together with the improved grading models, the next team's step is to improve comment models.