W207 Final Project:
Predicting English Essay Score

Kevin Demsich, Nicholas Lee, Harry Lu
W207: Applied Machine Learning / 2022 Fall / Section 3

Problem Motivation

Writing is a foundational skill. Sadly, it's one few students can hone, often because writing tasks are infrequently assigned in school. A rapidly growing student population, students learning English as a second language, known as English Language Learners (ELLs), are significantly affected by the lack of practice. While automated feedback tools make it easier for teachers to assign more writing tasks, they are not designed with ELLs in mind.

Existing tools cannot provide feedback based on the student's language proficiency, resulting in a final evaluation that may be skewed against the learner. Data science may be able to improve automated feedback tools to support the unique needs of these learners better.

We propose to build a model to facilitate this scoring process. A service that could automatically score students' essays would be our ultimate goal. Still, for this project, we aim to build a supervised machine learning model that can learn from current grading and use it to score unseen essays.

We found this project has significant potential value as it provides more resources for education. With this model, lecturers can save time on other high priorities such as class material design and live session preparation.

Dataset Description

The data contains one input field and five output fields. The input field is the text field, which is the essay. We ran some exploratory analysis on the input data and provided the summary below. The first column is the text_id as a unique identifier, and the second is the input column, the full text. After that, we have six rank-type metrics corresponding to each full text. The data is suitable for training both regression and classification models.

	text_id	full_text	cohesion	syntax	vocabulary	phraseology	grammar	conventions
0	0016926B079C	I think that students would benefit from learn...	3.5	3.5	3.0	3.0	4.0	3.0
1	0022683E9EA5	When a problem is a change you have to let it ...	2.5	2.5	3.0	2.0	2.0	2.5
2	00299B378633	Dear, Principal\n\nIf u change the school poli...	3.0	3.5	3.0	3.0	3.0	2.5
3	003885A45F42	The best time in life is when you become yours...	4.5	4.5	4.5	4.5	4.0	5.0
4	0049B1DF5CCC	Small act of kindness can impact in other peop...	2.5	3.0	3.0	3.0	2.5	2.5

Exploratory Data Analysis (finding relationships, creating metadata)

Before we built a model, the team studied the data, including some exploratory data analysis. We assume we can learn more about data or conduct feature engineering if we find some opportunities.

Input Data: Text Field

We know the average word count is 430, and the distribution ranges heavily between 200 and 600.

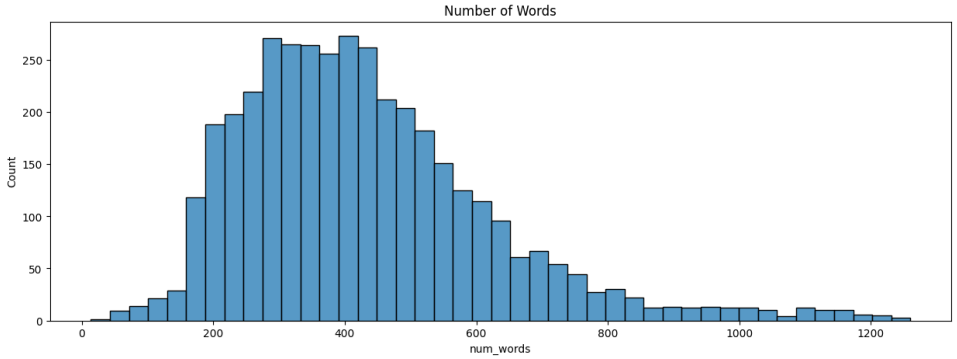
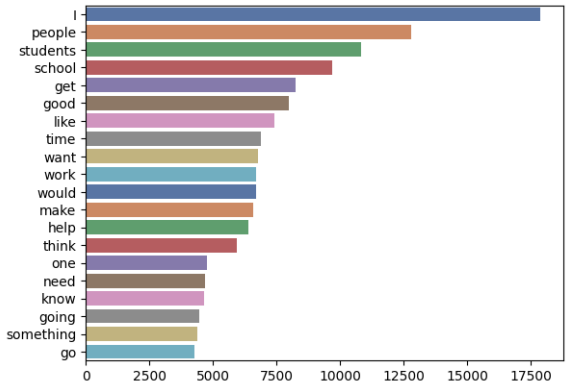


Figure 1: the average word counts of the input text field

We also ranked the top words used in the text, and the top 5 popular words are “I,” “people,” “student,” “school,” and “get.” We can also see the major topics discussed in the essays from the word cloud.



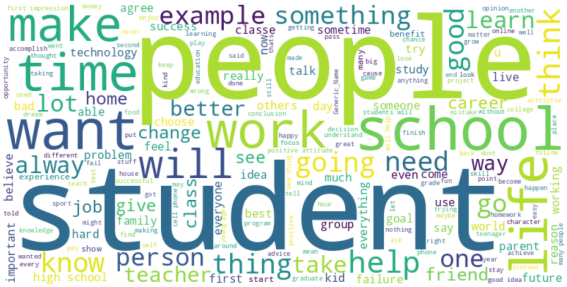


Figure 2: popular words from text field / Figure 3: word cloud produced from essays

Besides counting the occurrences of single words, we also explore the occurrences of a combination of words. This is a technique called **ngram**. For example, the top occurrence of the combination when ngram equals five are below:

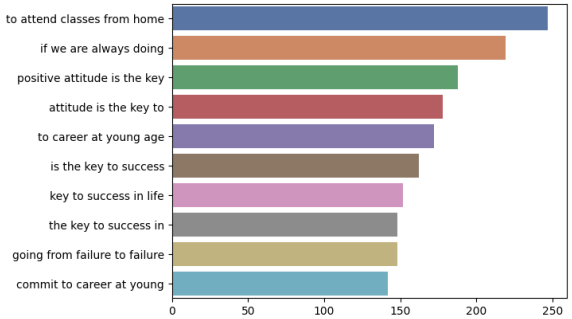


Figure 4: the top combinations (ngram=5)

With those EDA, we learned that (1) the scores are centered in 3, and the labels are imbalance, (2) the scores are positively correlated, (3) the average input length for us to decide the truncate threshold, and (4) the common words, phrase, etc.

We also used a pre-trained model BERTTopic to detect the topics from those essays. The result suggested the popular topics from the essays are, for example, people, life, online classes, school, and positive attitude.

	Topic	Count	Name
0	-1	338	-1_want_people_life_like
1	0	300	0_online_classes_home_students
2	1	226	1_hours_school_day_time
3	2	213	2_attitude_positive_life_good
4	3	204	3_accomplish_something_always_idle
5	4	179	4_technology_use_people_contact
6	5	156	5_years_high_graduate_early
7	6	154	6_career_young_age_high
8	7	142	7_decisions_decision_guidance_make
9	8	138	8_group_working_alone_work
10	9	129	9_park_play_fun_go
11	10	112	10_failure_success_enthusiasm_fail

Output Data: Scores

Each text sample has six output metrics that we need to predict:

1. Cohesion
2. Syntax
3. Vocabulary
4. Phraseology
5. Grammar
6. Conventions

These metrics are the scores evaluated from the essay and are on a discrete scale from 1 to 5 in 0.5 increments. Based on the output metrics, we found the distribution of the scores heavily centered on 3, especially for metrics like vocabulary. This will create a problem of “imbalanced labels” as we may not have enough labels of 1, 1.5, 4.5, and 5, and their corresponding essay. When there are not enough labels for the other classes, the model will acquire a bias to predict the most common class most of the time, as the other classes are seen too infrequently to be differentiated between.

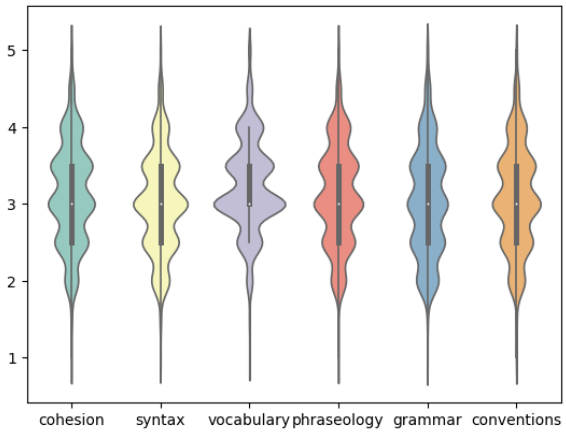


Figure 5: The distribution of the output metrics

We also checked the correlation between these scores and found that the scores positively correlated with each other.

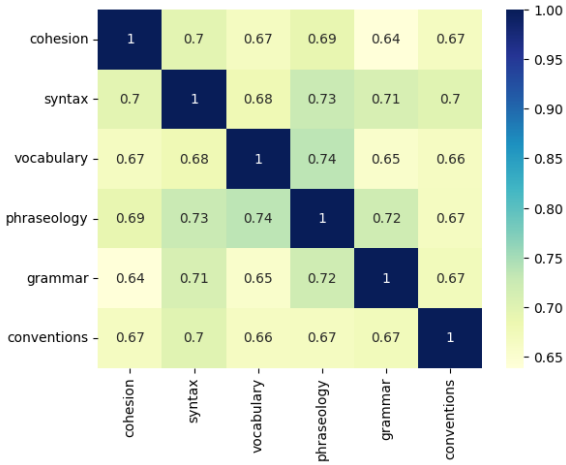


Figure 6: the correlation between each scoring metric

To illustrate the grouping of the average composite score of all six characteristics, we investigated the relationship between the highest score a student receives and their average score. We found there is a strong correlation between the highest score attained and the average composite score.

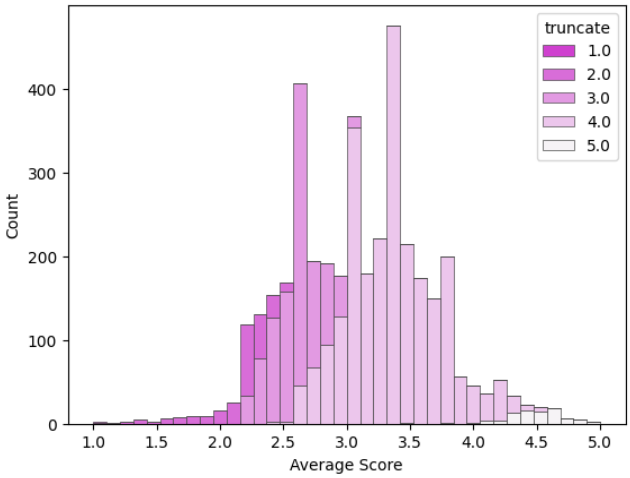


Figure 7: the relationship between the highest single score and the average composite score

ML Approaches

We plan to use embedding for this project. With the embedding, we can understand each word used in the essay and how “the occurrence of a word” matters in the score prediction.

Clean Feature, Data Preprocessing, and Train-Test Split

We first prepared the data by building customized text-cleaning functions. And before we started training, we ran a 60-40 train-test split to cut the data to 2,737 training samples and 1,174 testing samples. We also performed feature configurations, such as truncating the input to the desired length of 512 characters.

Build the model, leveraged on the pre-trained Bert model and LSTM layers

We used the pre-trained Bert Base Cased model to “learn” the essay and use the information from Bert for the downstream task. We then take the output from the Bert model to the LSTM layer, which can process not only single data points but also entire data sequences, including the text input we dealt with.

Train and Evaluate the Model

After we build the model, we will evaluate the model by calculating the accuracy of the prediction against the y output label.

Baseline Model

To have a baseline comparison, we assumed a baseline model called “always predict 3!” as 3 is the most frequent score in the dataset. With this model, the accuracy is calculated as 38%

Training

We set our model in the structure below. Starting with the TFBertModel, followed by LSTM layers and batch normalization layers. After the LSTM and normalization, we used two pooling layers and fed them to one output layer. The training took, on average, 1.5 hours to complete for each epoch. Due to the limited time and computing resources, we chose vocabulary score as the only output to predict before expanding the model to other outputs.

The below structure represents the model we use:

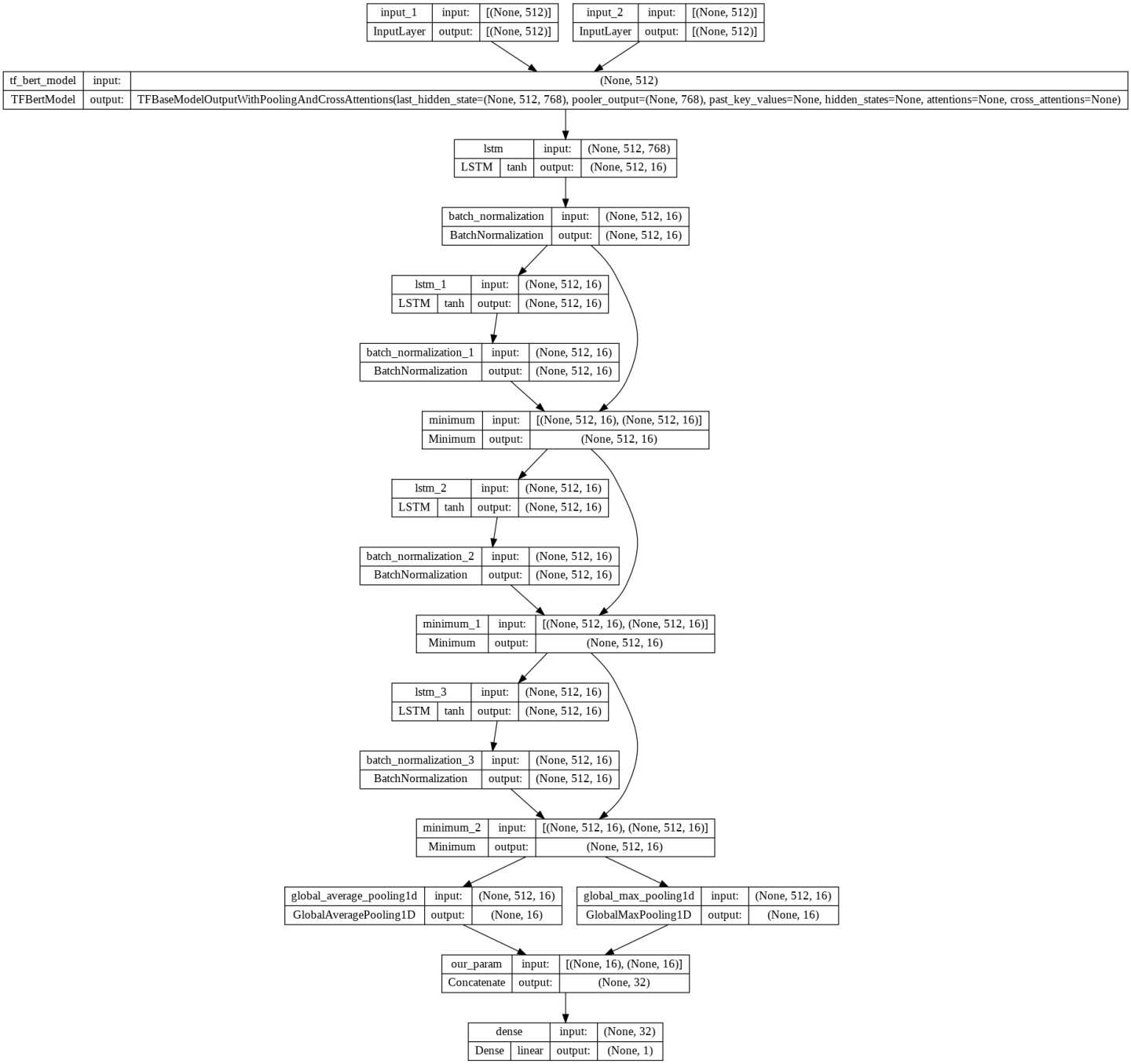


Figure 8: Model structure with layers, shape, and activate functions

Result

We learned 56,737 parameters in the model. The model provides an accuracy of 36%. But the distribution is more closer to the true label.

```
Model: "model_4"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_9 (InputLayer)           [(None, 512)]        0           []

 input_10 (InputLayer)          [(None, 512)]        0           []

 tf_bert_model_1 (TFBertModel)  TFBaseModelOutputWi  108310272   ['input_9[0][0]',
                                thPoolingAndCrossAt               'input_10[0][0]']
                                tentions(last_hidde
                                n_state=(None, 512,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 lstm_14 (LSTM)                 (None, 512, 16)      50240       ['tf_bert_model_1[0][0]']

 batch_normalization_14 (BatchN  (None, 512, 16)     64          ['lstm_14[0][0]']
 ormalization)

 lstm_15 (LSTM)                 (None, 512, 16)      2112        ['batch_normalization_14[0][0]']

 batch_normalization_15 (BatchN  (None, 512, 16)     64          ['lstm_15[0][0]']
 ormalization)

 minimum_10 (Minimum)           (None, 512, 16)      0           ['batch_normalization_14[0][0]',
                                                                  'batch_normalization_15[0][0]']

 lstm_16 (LSTM)                 (None, 512, 16)      2112        ['minimum_10[0][0]']

 batch_normalization_16 (BatchN  (None, 512, 16)     64          ['lstm_16[0][0]']
 ormalization)

 minimum_11 (Minimum)           (None, 512, 16)      0           ['minimum_10[0][0]',
                                                                  'batch_normalization_16[0][0]']

 lstm_17 (LSTM)                 (None, 512, 16)      2112        ['minimum_11[0][0]']

 batch_normalization_17 (BatchN  (None, 512, 16)     64          ['lstm_17[0][0]']
 ormalization)

 minimum_12 (Minimum)           (None, 512, 16)      0           ['minimum_11[0][0]',
                                                                  'batch_normalization_17[0][0]']

 global_average_pooling1d_4 (Gl  (None, 16)          0           ['minimum_12[0][0]']
 obalAveragePooling1D)

 global_max_pooling1d_4 (Global  (None, 16)          0           ['minimum_12[0][0]']
 MaxPooling1D)

 our_param (Concatenate)        (None, 32)           0           ['global_average_pooling1d_4[0][0
                                                                 ]',
 dense_4 (Dense)                (None, 1)            33          ['our_param[0][0]']

==================================================================================================
Total params: 108,367,137
Trainable params: 56,737
Non-trainable params: 108,310,400
__________________________________________________________________________________________________
```

Figure 9: the model summary

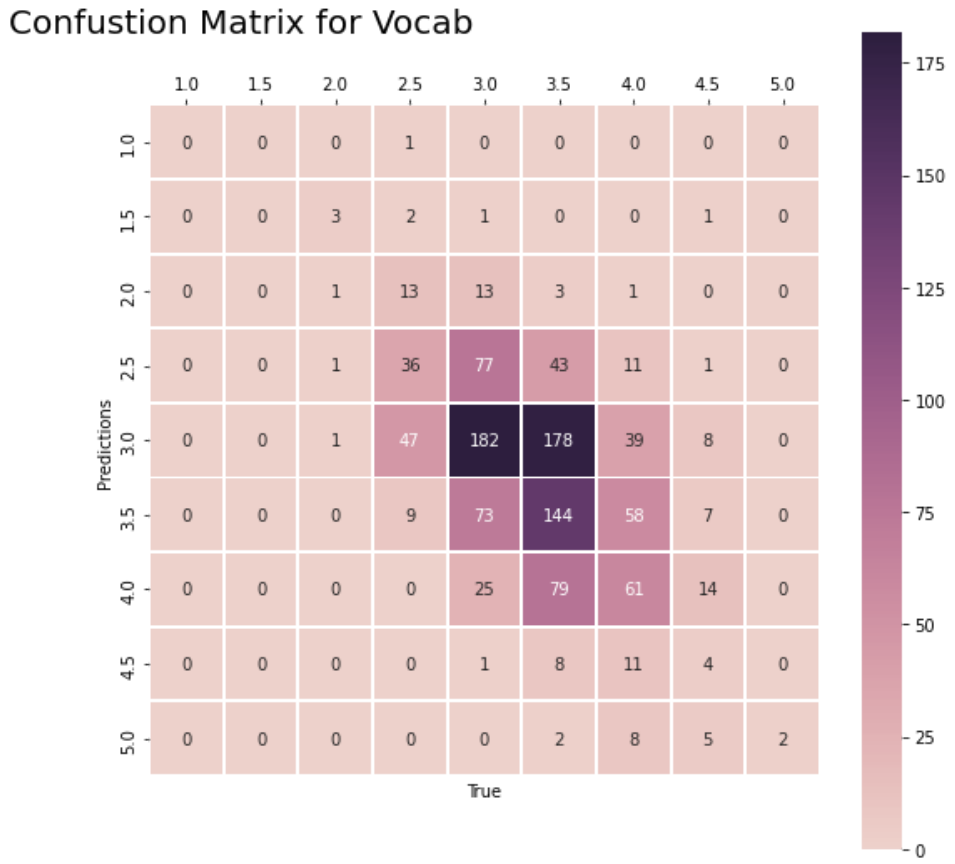


Figure 10: the Confusion Metrix for Vocabulary Score

Conclusion

Our model predicted the vocabulary metric's scores with 36% accuracy but better distribution on the prediction label. We conclude that this model can potentially help future iterations for other output metrics and meet the goal of reducing the time of human review on essays. We also see some opportunities for improvement, which we will list in the Future Work session.

From the Kaggle competition, the highest score attained was 43.3% across all 6 output metrics. It appears that having more training examples from the near edge-distribution classes would greatly improve the model results for all participants in the study. Data augmentation approaches are limited in improving model performance without introducing too much bias or overfitting to known examples. Although these results are currently unsuitable for a fully automated system to facilitate grading, this experiment lays the groundwork for promising future research into this sector.

Future Work

In the future work session, we want to highlight some of our progress and plan to explore more as a future work plan.

Imbalance Data

From the exploratory data analysis, we know the output label is centered about 3 and has very few labels for extreme scores, such as 1 or 5. Therefore, this might cause some issues due to imbalanced data. We created the synthesized sample to minimize the effect to solve the data imbalance issue. The approach we took is oversampling the data, as we do not want to lose data if we choose to undersampling. Given the few

examples, we had for edge scores, using undersampling would restrict all other classes for which we have many responses to be limited to equal quantities. On the other hand, using oversampling, we could generate new examples from the limited set of examples we had on the edge cases to build up an equal pool of essays for the model to learn from.

Logistically, we produced the expanded data set before training the machine learning model. Several methods are available to oversample a dataset used in a typical classification problem. And the approach we chose is the Synthetic Minority Oversampling Technique or SMOTE for short. Leveraging the SMOTE technique, we synthesize more samples to balance the under-represented labels [1, 1.5, 2, 4, 4.5, 5]. We added 1000 examples for each underrepresented class as proof of concept. With this, we alleviated the imbalance issue for our model, aiming to produce a more robust learning effect and improve overall predictions. After several model iterations, the team leaned towards a regression approach instead of classification. Hence, we did not use the augmented extra data, anticipating those could diverge the prediction result.

Classification vs. Regression

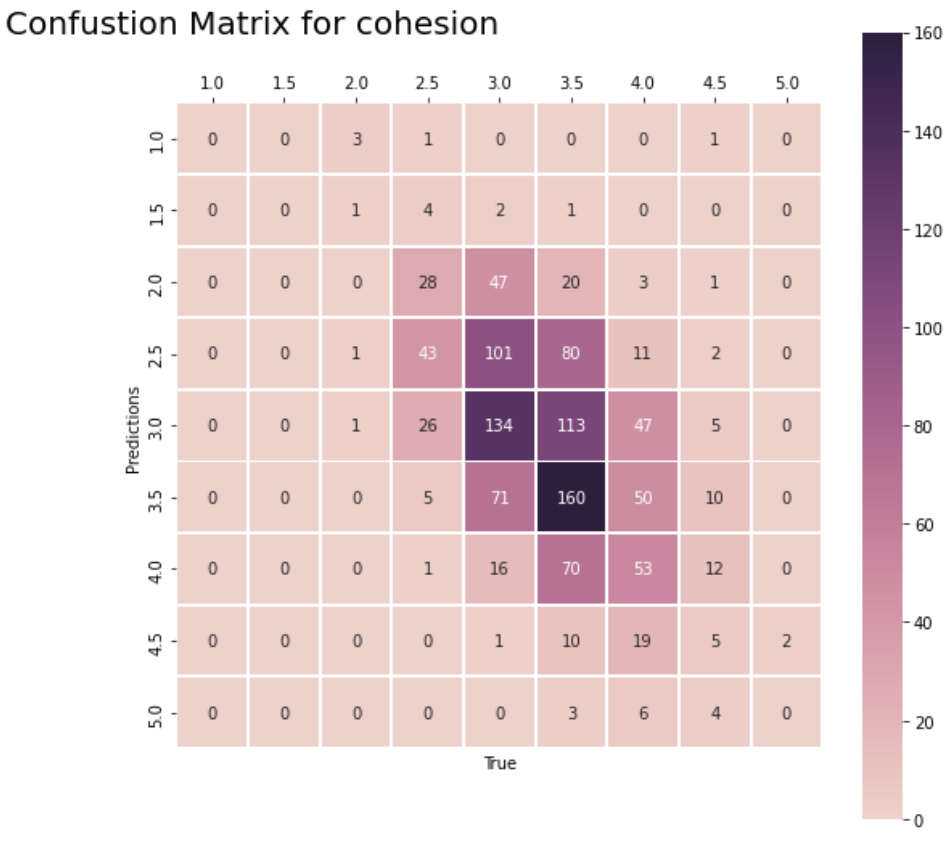
During this project, we have tried to define this problem in two ways: classification and regression. Due to the rank type output label, we considered both classification and regression approaches. Although we finalized the regression method, we want to further explore this question with a classification approach. Both methods have pros and cons listed below:

1. Classification:
    a. Pros: We have a fixed number of nine outputs.
    b. Cons: the punishment for predicting a 1 to a 5 is the same as predicting a 4.5 to a 5, as they both count as inaccurate predictions. In reality, the difference between a 4.5 to a 5 is very different from the difficulty of going from a 4.5 to a 5.
2. Regression:
    a. Pros: The output score is a numeric type, and we can calculate how “off” our predictions are via standard regression loss function (i.e., MSE)
    b. Cons: The rank-type data is not linear, and the scores are discrete and not continuous. The distance between 4.5 and 5 differs from the distance between 1 and 1.5

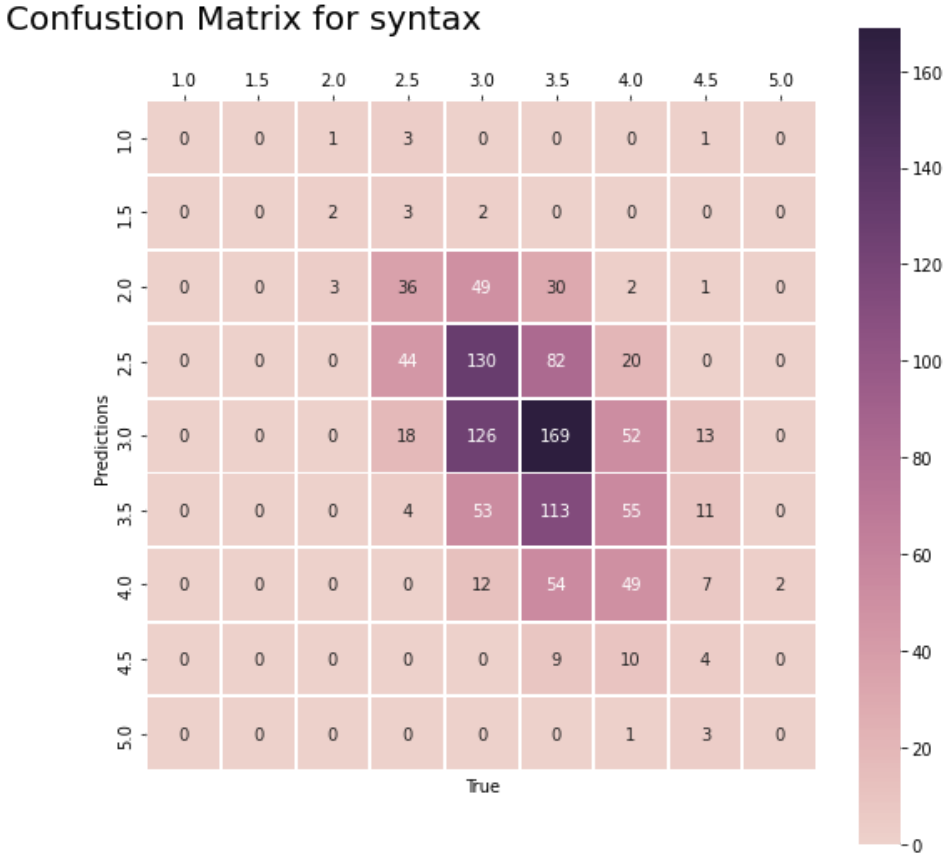
Appendix

1. Data: link
2. Notebooks
    a. On GitHub
        i. Final Notebook(s)
            1. Final Model and Analysis
        ii. Test Notebook(s)
            1. Harry Notebook (test model)
            2. Harry Notebook (feature engineering)
            3. Nick Notebook
    b. On DeepNote
        i. EDA (Harry)
        ii. EDA (Nick)
        iii. EDA (Kevin)
3. Reference:
    a. How to deal with Imbalance data in classification
    b. Document of bert-base-cased Model

4. Supplemental Figures:

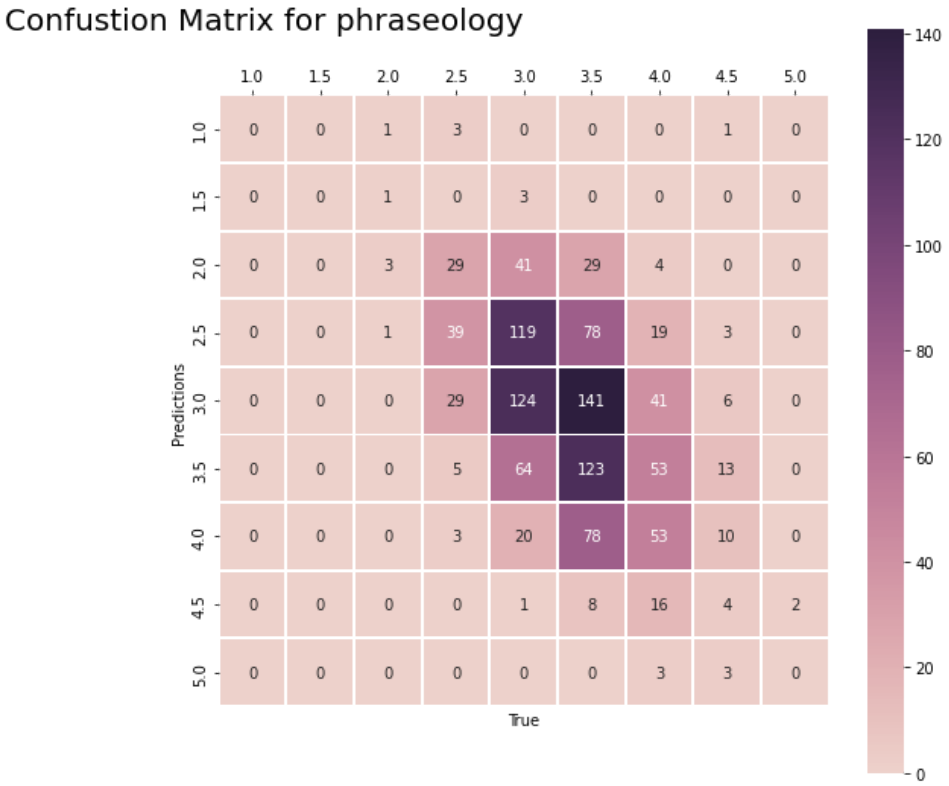


a.

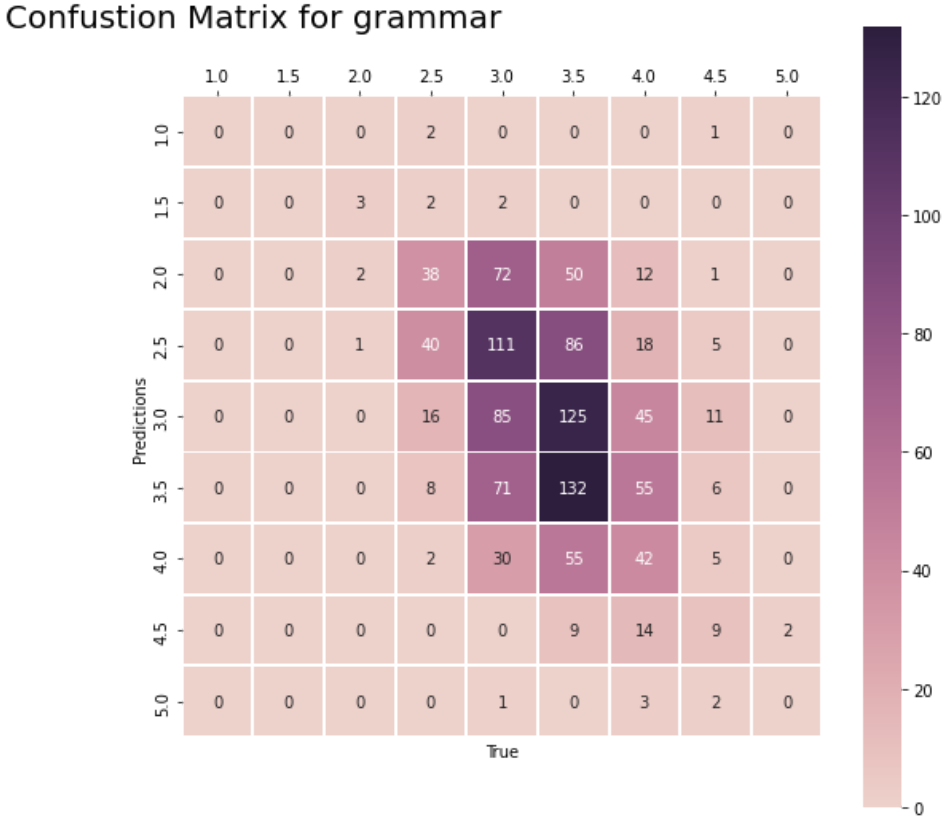


b.

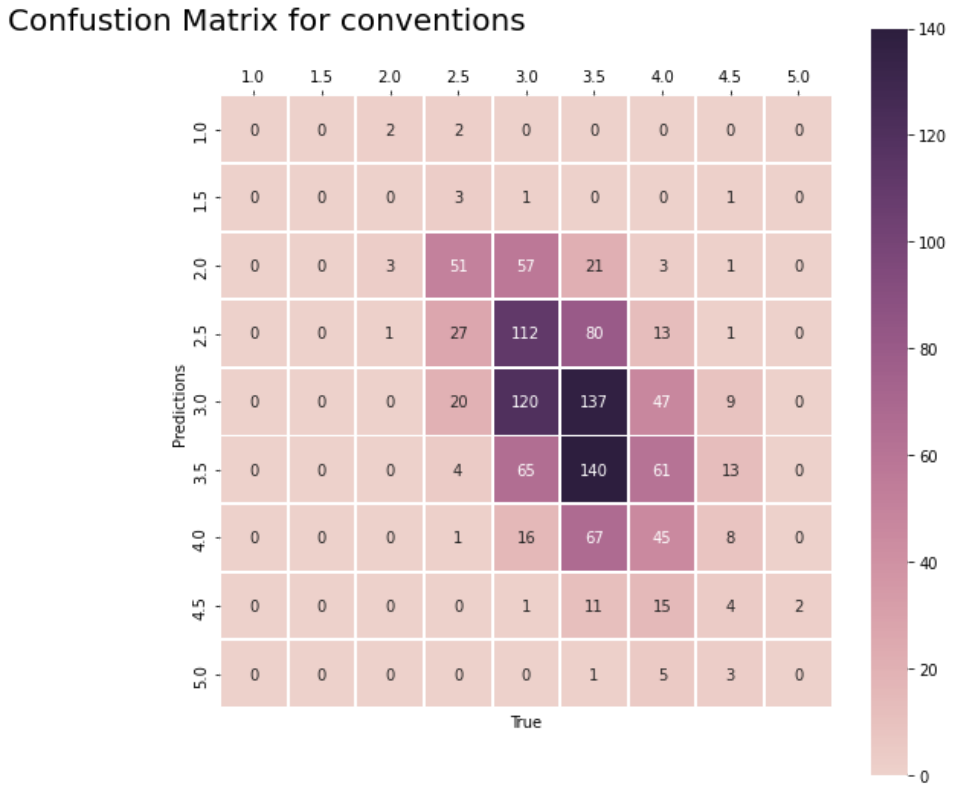
c.


Confustion Matrix for phraseology
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
1.0 0 0 1 3 0 0 0 1 0
1.5 0 0 1 0 3 0 0 0 0
2.0 0 0 3 29 41 29 4 0 0
2.5 0 0 1 39 119 78 19 3 0
3.0 0 0 0 29 124 141 41 6 0
3.5 0 0 0 5 64 123 53 13 0
4.0 0 0 0 3 20 78 53 10 0
4.5 0 0 0 0 1 8 16 4 2
5.0 0 0 0 0 0 0 3 3 0
Predictions
True
140 120 100 80 60 40 20 0

d.


Confustion Matrix for grammar
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
1.0 0 0 0 2 0 0 0 1 0
1.5 0 0 3 2 2 0 0 0 0
2.0 0 0 2 38 72 50 12 1 0
2.5 0 0 1 40 111 86 18 5 0
3.0 0 0 0 16 85 125 45 11 0
3.5 0 0 0 8 71 132 55 6 0
4.0 0 0 0 2 30 55 42 5 0
4.5 0 0 0 0 0 9 14 9 2
5.0 0 0 0 0 1 0 3 2 0
Predictions
True
120 100 80 60 40 20 0

e.


Confustion Matrix for conventions
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
1.0 0 0 2 2 0 0 0 0 0
1.5 0 0 0 3 1 0 0 1 0
2.0 0 0 3 51 57 21 3 1 0
2.5 0 0 1 27 112 80 13 1 0
3.0 0 0 0 20 120 137 47 9 0
3.5 0 0 0 4 65 140 61 13 0
4.0 0 0 0 1 16 67 45 8 0
4.5 0 0 0 0 1 11 15 4 2
5.0 0 0 0 0 0 1 5 3 0
Predictions
True
140
120
100
80
60
40
20
0