CommonLit Readability Prize

1. Introduction

The cultivation of reading ability plays a key role in language learning. Text readability, which has been formally defined as the sum of all the elements in text material that affect readers' understanding, reading speed, and level of interest, is influenced by multiple variables [1]. These variables may include the author's writing style, the reader's education background, the structure and logic of the article, etc. A method that can rate the complexity passages is helpful. For example, teachers can assign articles with appropriate readability to students to improve their reading skills. Manually labelling the readability and complexity of texts is one of the methods for evaluation. However, this way is too time-consuming and cannot be standardized. Therefore, an algorithm that can automatically evaluate the readability of the text would be more convenient.

The goal of this project is to recognize the complexity of a piece of text by applying artificial intelligence skills. Submissions for this competition are scored on the root mean squared error (RMSE). Two datasets are used in this project: Competition Dataset and Augmented Dataset. Competition Dataset is provided by the Kaggle, which contains excerpts from different articles. As for the Augmented Dataset, it contains more content from the original article rather than Competition Dataset. In this project, we have tried and explored a variety of methods.

For the part of Machine Learning methods, the embeddings extracted by spaCY and some readability metrics such as Fleisch score are concatenated as the input features for some regression models. Though the training time is fast, and this method is easy to implement, the accuracy is not high enough.

For the part of Deep Learning methods, we have tried to compare the performance between BERT and RoBERTa models. The result demonstrates that RoBERTa model could get a better score in this competition. We have trained a total amount of 19 models until now and named them from model 1 to model 19. Then we employed a scheme that implements a stacking ensemble of model 4 and model 19. Model 4 is a RoBERTa-base model, pre-trained with Competition Dataset, while model 19 is a RoBERTa-large model, pre-trained with Augmented Dataset. In the fine-tuning stage, 5-fold cross-validation is applied. To be specific, we will have 5 models after the fine-tuning, each one is trained by 4 folds and use the fold that does not participate in training to do evaluation. Next, two levels are designed for the stacking ensemble. At the base level, the RoBERTa-base and RoBERTa-large model with the highest RMSE score is chosen. At the meta-level, Ridge is used as the Regressor.

It is worth mentioning that all models we trained do not perform well in the 4th fold. To deal with this problem, we proposed a new model that fits well with the content of training data, which extracts the embeddings from RoBERTa-large and applies SVM for regression tasks. Then the prediction of this model is used to blend the 4th fold in all models at the base level. Our goal is to ensure that our final ensemble predictions are slightly better but not over-fitting. From the results, we can see that the method works and gets the highest score in many experiments.

Consequently, our contribution to this project is reflected in many aspects. Firstly, we have implemented an efficient method that can accurately evaluate the readability of the text. Secondly, we have explored some other methods that can serve as a good reference. The final RMSE score of our team is 0.457, which ranked 88th in the competition and won the silver medal.

2. Related work

Currently, some educational texts are matched to readers using commercially available formulas. However, it has some issues. For example, the Lexile, can be cost-prohibitive, lack suitable validation studies, and suffer from transparency issues when the formula's features aren't publicly available.

Others using traditional readability methods, such as Gunning fog formula, McLaughlin's SMOG formula, Flesch Scores, etc, also have their issues. The Flesch-Kincaid index [2] is based on weak proxies of text decoding (i.e., characters or syllables per word) and syntactic complexity (i.e., number of words per sentence). As a result, they lack construct and theoretical validity. These methods based on traditional features can be effective in some specific fields, but they are difficult to be universally applied to larger texts.

In addition to the above features, existing machine learning methods also use statistics to integrate features, such as the number of sentences per text, average and maximum number of words per sentence, the average number of characters per word, etc. In addition, some others are also added into features, such as the Discourse cohesion features [3], the Lexico-semantic features [4], etc., using linear functions such as SVM for regression prediction [5]. These methods still have shortcomings in the order of sentences and structure of how sentences are constructed.

A lot of work has been done on the application of neural network models for sentiment analysis or text classification tasks, but less attention has been paid to text readability evaluation tasks. The BERT model, based on the multi-layer bidirectional Transformer encoder, is different from other recent language representation models. BERT model, which uses transformer encoder blocks with multi-head self-attention, had achieved stat-of-the-art in 12 NLP tasks including text classification. Therefore, the BERT model can be well applied to the task of text readability evaluation [6]. However, the BERT model still has some shortcomings in pre-training stage, such as the use of static masks in MLM tasks, and the pre-training effect of the NSP task is not very satisfactory. A network called ReadNet was proposed [7], which is a comprehensive readability classification framework using a hierarchical converter network. The self-attention part of the transformer encoder can better model the remote and global dependencies between words. It achieved good results in text readability evaluation.

3. Methods

3.1 Machine Learning

3.1.1 Features

In feature construction, we first applied some traditional statistical-based feature extraction methods, such as the number of characters in each text, the number of words, the number of sentences, and the ratio of the number of words to the number of characters. In addition, the number of words containing the syllable 'aeiouy' in the text is also counted, as well as the number of unique words in the text and its ratio to the number of words which is regarded as text diversity.

Since it is difficult to include more text information in statistical feature extraction, we also selected two Fleisch score evaluation scores as features. The calculation formulas are as follows,

$$fleisch\ score1 = 206.835 - 1.015 * \frac{number\ of\ words}{number\ of\ sentences} - 84.6 * \frac{number\ of\ syllables}{number\ of\ words} \quad (1)$$

$$fleisch\ score2 = 0.39 * \frac{number\ of\ words}{number\ of\ sentences} + 11.8 * \frac{number\ of\ syllables}{number\ of\ words} - 15.59 \quad (2)$$

The relationship between these two features and target is shown in the figures below:

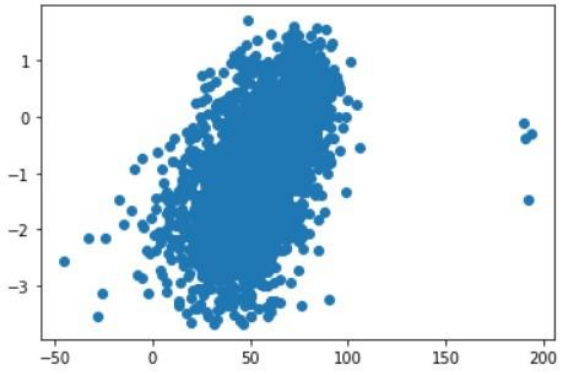


Figure 1. Fleisch score1

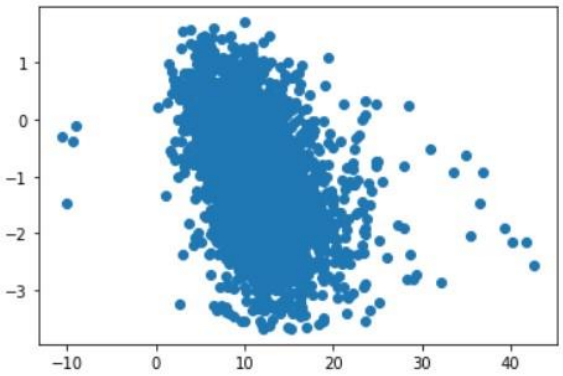


Figure 2. Fleisch score2

In addition to the features selected above, we also applied the spaCy library for feature extraction. spaCy is a module for NLP tasks, which can realize functions such as word segmentation, part-of-speech tagging, named entity recognition, and dependency analysis. We loaded the 'en_core_web_lg' in spaCy to extract features of the text.
The above feature combination is used as the input feature of the model.

3.1.2 Regression Models

Regarding the construction of the regression model, we tried and tested a variety of models, including linear regression, Ridge, Lasso, SVR, RandomForest, Adaboost, etc. Then we compared these models with 10 fold cross-validation. The model parameters are set as follows:

model	**parameters**
Linear regression	auto
Ridge	alpha=1
Lasso	alpha=0.5, normalize=True
SVR	kernel='rbf', gamma='auto'
RandomForest	n_estimators=100
Adaboost	n_estimators=120

Table 1 the model parameters of Machine Learning

Compared with some existing neural networks, the biggest advantage of traditional machine learning methods is that they run fast. And their feature extraction and model construction are relatively simple. However, because feature extraction cannot contain more text information, the prediction accuracy of the model may be insufficient. In addition, due to the relatively small training data set in this project, it is difficult for the machine learning model to achieve better training results. In this regard, compared with the BERT model based on a huge corpus pre-training, it has a great disadvantage.

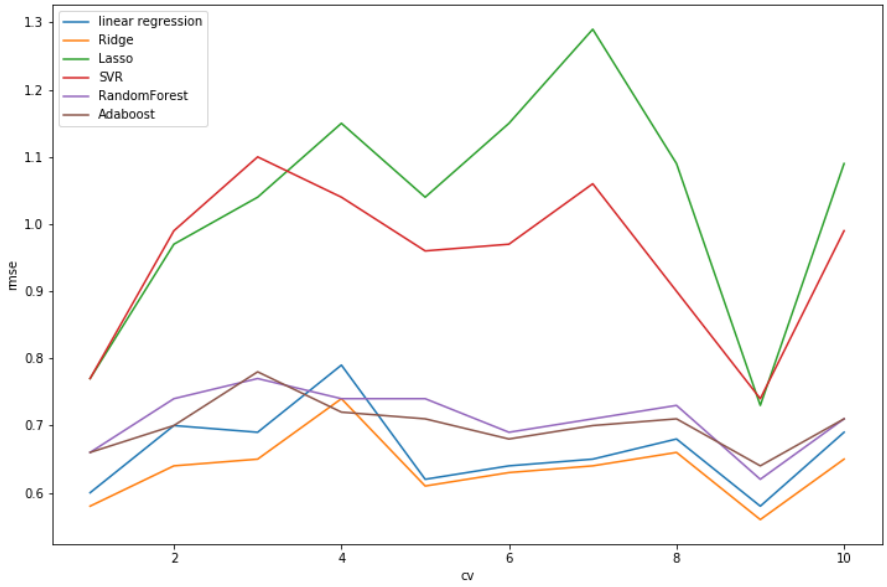


Figure 3 the comparison of different regression model

3.2 RoBERTa-base

3.2.1 Bert vs RoBERTa-base

Considering the drawbacks of machine learning methods, we choose RoBERTa, a transformer-based model, which is better than BERT in many aspects to be our features encoder.

RoBERTa is a transformer model pre-trained in a self-supervised manner using masked language modeling (like BERT) targets on a large amount of English data (including BookCorpus, English Wikipedia, CC-News, and OpenWebText). This means that it is only pre-trained on the original text, and no one tags them in any way through an automatic process to generate input and labels from these texts.

Specifically, RoBERTa is trained with dynamic masking, which solves the problem that some masked words in the pre-training stage of BERT might be unseen in the fine-tuning stage. Also, compared with BERT, RoBERTa uses full sentences without NSP loss, large mini-batches, and a larger byte-level BPE.[8] The improvement of RoBERTa compared to BERT are shown below:

1. **RoBERTa optimizes the model details of general BERT**

BERT is optimized with Adam using the following parameters: $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\varrho = 1e\text{-}6$ and L2 weight decay of 0.01.[8] As RoBERTa chooses to use a larger batch size, it changes $\beta 2$ to 0.98.

2. **RoBERTa optimizes the general BERT training strategy**

RoBERTa uses dynamic masking instead of original static masking. The mask is not executed during preprocessing but is dynamically generated every time a sequence is provided to the model RoBERTa is trained with large batches(8K), which improves perplexity for the masked language modeling objective, as well as end-task accuracy.[1]

3. **RoBERTa optimizes the model at the data level**

RoBERTa uses larger training data set. It upgrades the 16G data set to a 160G data set and changes multiple steps to find the best hyperparameters.

RoBERTa has been proved to be a better pre-trained model than BERT in many works. We download RoBERTa -base-uncased and BERT-base-uncased from the Hugging face website and decided to run some experiments to verify this statement. The default config setting and comparison result will show in 3.2.2.

3.2.2 Fine-tuning RoBERTa-base

The process of fine tuning is to initialize a network with the trained parameters, which can be obtained from the trained model. Then we use competition dataset to train this network. The parameter adjustment method is as same as the from scratch training process.

We need to re-learn the last layer and have a relatively large learning rate compared with other layers. This effectively utilizes the powerful generalization ability of the model. And it also eliminates the need for complex models and time-consuming training. So fine tuning is an appropriate choice when the amount of data is insufficient.

We use one Attention-head layer and one regression layer to do the finetune. The input of the Attention-head layer is the output of the RoBERTa layer. The purpose of using this layer is to gain a stronger feature representation from the input sentences. The formulas of the Attention layer are shown below:

$$f = V(tanh\,(W(x))) \quad (3)$$

$$\alpha = softmax(f) \quad (4)$$

$$c = sum(\alpha * x, dim = 1) \quad (5)$$

$$output = W_{out}\big(dropout(c)\big) \quad (6)$$

As we can see from the formula (3), the x is denoted as the output of the RoBERTa (input dim size is either 768 or 1024), W is denoted as a linear layer with 512 hidden sizes. tanh as the activation function. V is denoted as a linear layer with 1 hidden size. f is a score function, and its shape is (batch sizes, seq_len, 1). In formula (4), we compute an attention weight α by using SoftMax function whose shape is the same as f. In formula (5), we multiply attention weight α by x and do the summation on dim 1 (horizontally compress the dimension to 1) to get the context vector c. In this way, based on the attention mechanism, some features that the model thinks are important might be "highlight". In the last step, $dropout$ is a method to lower the probability of getting overfitting. W_{out} is a regression layer with output feature dim = 1. After these 4 steps, we have done the main part of finetuning stage.

In this competition, provided training data is used to fit RoBERTa model. The training dataset is divided into 5 folds, the loss and RMSE on each fold in 35 epochs is shown as follows.

There is the setting of these two models.

Parameter	**value**
batchsize	16
Max_len	248
learning rate	1e-7
SEED	1000
fold	5
loss	RMSE
Pre-train	True
Fine-tune approaches	Attention-head layer + regression layer
GPU resource	Nvidia RTX 3090 x 2
Optimizer	AdamW

Table 2 the setting of BERT-base-uncased and RoBERTa-base-uncansed

The results of these experiments are shown as below:

RMSE	BERT-base-uncased	RoBERTa -base-uncased
No Pretrain + finetune fold1	0.5029187449263255	0.4802685083486929
No Pretrain + finetune fold2	0.4865362226687062	0.46584051877659405
No Pretrain + finetune fold3	0.4890528672329882	0.47890673274629275
No Pretrain + finetune fold4	0.4972185440877818	0.47880262303135024
No Pretrain + finetune fold5	0.4775155027375212	0.47573807270278945
Average scores	0.4906483763306646	0.4759112911211439

Table 3 the comparison of BERT-base-uncased and RoBERTa-base-uncased without pre-training

Here we compare the distribution from the labels and the predicted values, in a perfect scenario they should align. But it can be seen from the image, in the range between -0.6 and 0.2, the distribution of label and prediction are discrete.

The distribution of prediction in this range is much higher than that of the label, which means RoBERTa made some mistakes when dealing with cases which target in this interval.

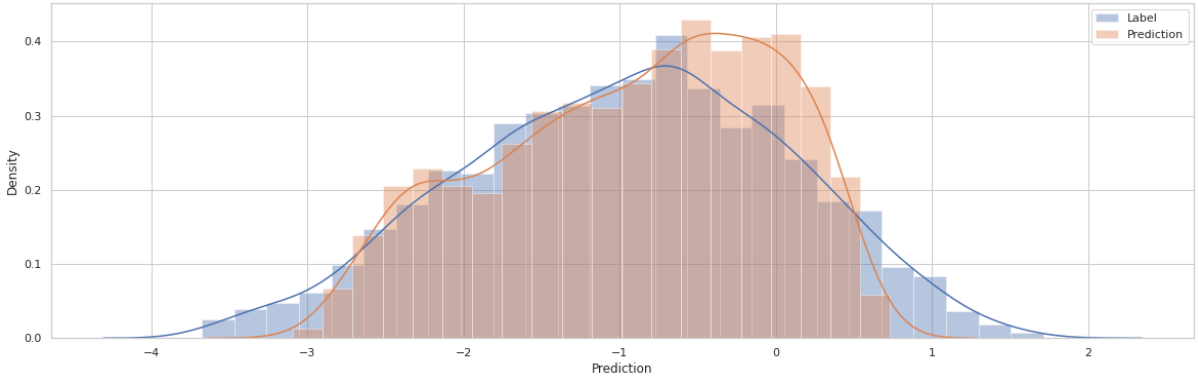


Figure 4 the distribution from labels and predictions of RoBERTa-base-uncased

To analysis the error, some cases with target labels between -0.6 and 0.2 are chosen as examples to test RoBERTa-base.

ID	Text	Target
CASE1	My hope lay in Jack's promise that he would keep a bright light burning in the upper story t o guide me on my course. On a clear night this light was visible from the village, but some how or other I failed to take into account the state of the weather. The air was full of eddyin g flakes, which would render the headlight of a locomotive invisible a hundred yards distan t. Strange that this important fact never occurred to me until I was fully a fourth of a mile fr om the village. Then, after looking in vain for the beacon light, the danger of my situation s truck me, and I halted. "I am certain to go wrong," I said to myself. "It is out of my power to follow a direct course without something to serve as a compass. I will go back to the village and wait till morning."	-0.384126

CASE2	It was a bright and cheerful scene that greeted the eyes of Captain Raymond and his son as they entered the parlor of the adjacent cottage. It was strictly a family gathering, yet the room was quite full. Mr. Dinsmore was there with his wife, his daughter Elsie and her children, Edward and Zoe, Elsie Leland with her husband and babe, Violet Raymond with her husband's two little girls, Lulu and Grace, and lastly Rosie and Walter. Everybody had a kindly greeting for the captain, and Violet's bright face grew still brighter as she made room for him on the sofa by her side. "We were beginning to wonder what was keeping you," she said. "Yes, I'm afraid I am rather behind time," he returned. "I hope you have not delayed your tea for me, Mrs. Dinsmore." "No; it is but just ready," she said. "Ah, there's the bell. Please, all of you walk out." When the meal was over all returned to the parlor, where they spent the next hour in desultory chat.	-0.417479
CASE3	Milka and John are playing in the garden. Her little sister is playing too. Milka is ready to start classes next week and it will be her first term in school. In the morning, Milka gets up early to take a bath. She puts on her school uniform and carries her school bag. Her Mother gives her two thousand shillings for school fees and five hundred shillings for transport. Then, she quickly goes to school. Meanwhile, her big brother stays at home. He is still in his bed and sleeps. Once she grows up and graduates school, Milka dreams to build a beautiful house for her and her family. While she is at school, she is very active and participates in all the activities. The teachers love her attitude. Milka listens carefully to her teacher. Her classmates admire her too, because she is a kind girl. At break time she tries to help other classmates with their practical exercies and homeworks.	0.308700
CASE4	Debugging is the process of finding and resolving of defects that prevent correct operation of computer software or a system. Debugging tends to be harder when various subsystems are tightly coupled, as changes in one may cause bugs to emerge in another. Numerous books have been written about debugging, as it involves numerous aspects, including interactive debugging, control flow, integration testing, log files, monitoring (application, system), memory dumps, profiling, Statistical Process Control, and special design tactics to improve detection while simplifying changes. The terms "bug" and "debugging" are popularly attributed to Admiral Grace Hopper in the 1940s. While she was working on a Mark II Computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system.	-1.919389

It is obvious that cases with target label in this range has more spoken dialogue fragment. To improve the performance of RoBERTa on these datasets, pretrain of RoBERTa is required.

3.2.3 Pretrained on competition dataset RoBERTa-base

Transformer models like RoBERTa trained on a huge amount of data on an MLM (masked language modeling) task is called pre-training.

To improve the performance of RoBERTa-base on the competition dataset, the original pre-trained RoBERTa is further pre-trained to predict masked words. This means that the original RoBERTa now has some more context of the competition data, which helps next fine-tuning for the prediction of the score.

To pre-train RoBERTa-base, DataCollatorForLanguageModeling creates masked data from all the text that is available in the 'excerpt' column. For example, is original text is “Math is an interesting subject.”, after masking the masked text should be "[MASK] is an interesting subject.”

The steps of pre-training are shown as following:

Step	Training Loss	Validation Loss	Runtime	Samples Per Second
200	No log	1.441347	32.183100	88.276000
400	No log	1.383025	32.088000	88.538000
600	1.550100	1.340914	32.012800	88.746000
800	1.550100	1.319844	32.065800	88.599000

Table 4 the steps of pre-training

The common approach for using the pre-trained BERT model is to replace the original output layer with a new task-specific layer and fine-tune the complete model.[9]

We use a pre-trained RoBERTa-base as the source model. Then create a new model, the target model. It replicates all model designs and their parameters on the source model except the output layer. We assume that these model parameters contain the knowledge learned on the source data set. And this knowledge is also applicable to the target data set. We also assume that the output layer of the source model is closely related to the label of the source data set. So, it is not used in the target model. By adding an output layer whose output size is the number of target data set categories to the target model, and initializing the model parameters of this layer randomly, we can train the target model on the target data set. We will train the output layer from scratch. And the parameters of the remaining layers are all fine-tuned based on the parameters of the source model.

Training data is divided into 5 folds, RMSE of model finetuned on each fold is shown as following:

RMSE	BERT-base-uncased	RoBERTa -base-uncased
Pretrain + finetune fold1	0.5029187449263255	0.4803272432266614
Pretrain + finetune fold2	0.4865362226687062	0.4657474895163564
Pretrain + finetune fold3	0.4890528672329882	0.4825870128395259
Pretrain + finetune fold4	0.4972185440877818	0.4813479930638911
Pretrain + finetune fold5	0.4775155027375212	0.47843833162094274
Average scores	0.4906483763306646	0.477

Table 5 the RMSE of RoBERTa-base with pre-training

The final prediction of test dataset is the mean prediction value of 5 finetuned model. The distribution from the labels and the predicted values are align on training dataset.

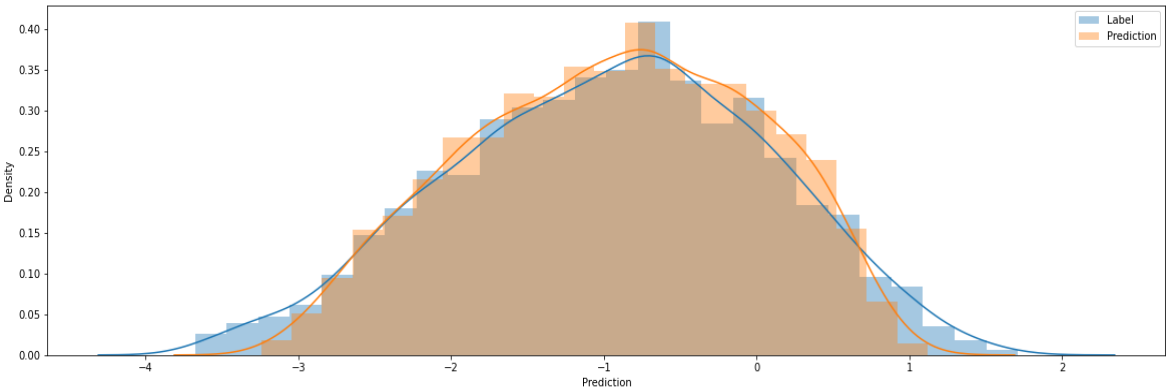


Figure 5 the distribution from the labels and the predictions of RoBERTa-base with pre-training

3.3 RoBERTa-large

3.2.1 RoBERTa-base vs RoBERTa-large

Compared with RoBERTa-base, RoBERTa-large has more layers and larger dimension. It has more parameters, which cost more memory and time to train and predict. RoBERTa-large has better performance on test dataset. The difference between RoBERTa-base and RoBERTa-large is shown as following:

	RoBERTa-base	RoBERTa-large
L (number of network layers)	12,	24
H (dimension of hidden layer)	768	1024
Total Parameters	110M	340M
Use GPU memory	more than 7G	more than 32G

Table 6 the difference between RoBERTa-base and RoBERTa-large

3.4 3.2.2 fine-tuning RoBERTa-large

The approach of fine-tuning of RoBERTa-large is as same as that of RoBERTa-base (3.2.2). We divide training dataset into 5 folds, and fit RoBERTa-large to each fold. We get RMSE of each fold when training:

RMSE	RoBERTa-large
No Pretrain + finetune fold1	0.4850311336319142
No Pretrain + finetune fold2	0.4697397578110472
No Pretrain + finetune fold3	0.4861952606785206
No Pretrain + finetune fold4	0.5157311407111126
No Pretrain + finetune fold5	0.4478779947540768
Average scores	0.4804

Table 7 the RMSE of RoBERTa-large without pre-training

The distribution from the labels and the predicted values on training dataset are close to coincide. But there are still error predictions in range [-1.4, 2].

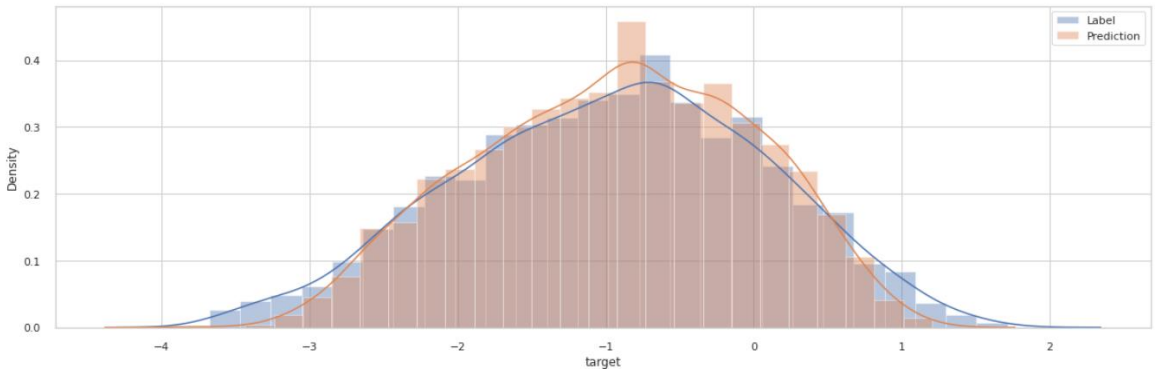
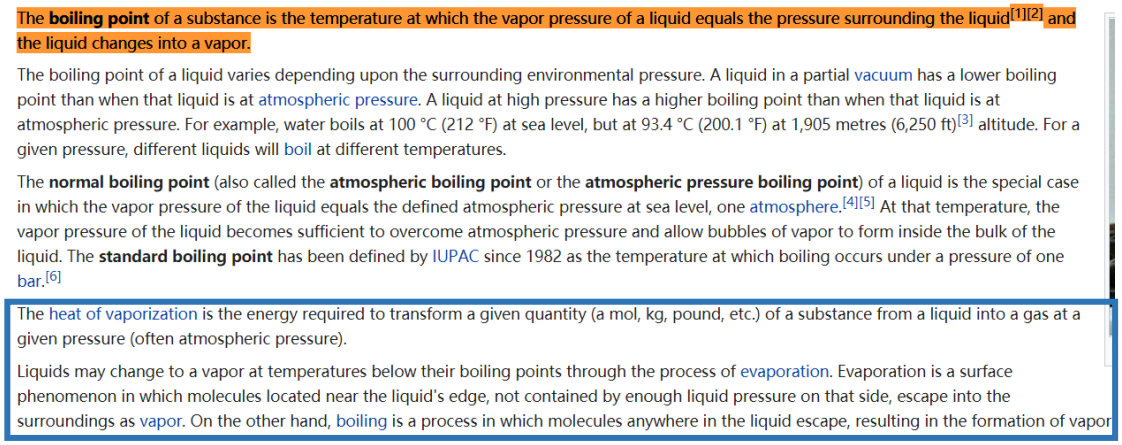


Figure 6 the distribution from the labels and the predictions of RoBERTa-large without pre-training

3.3.3 Pretrained on Augmented Dataset RoBERTa-large

Considering Competition Dataset is limit, we use URL legal in data as source to form Augmented Dataset. When pretrain RoBERTa-large, Augmented Dataset is trained for its MLM (masked language modeling). For example, case “2b2fdfc8c” has an URL link https://en.wikipedia.org/wiki/Boiling_point.

On this website the start of the case content is highlighted. The text near to the case content is chosen to add to Augmented Dataset, which augments the pre-training data set.


The **boiling point** of a substance is the temperature at which the vapor pressure of a liquid equals the pressure surrounding the liquid[1][2] and the liquid changes into a vapor.

The boiling point of a liquid varies depending upon the surrounding environmental pressure. A liquid in a partial vacuum has a lower boiling point than when that liquid is at atmospheric pressure. A liquid at high pressure has a higher boiling point than when that liquid is at atmospheric pressure. For example, water boils at 100 °C (212 °F) at sea level, but at 93.4 °C (200.1 °F) at 1,905 metres (6,250 ft)[3] altitude. For a given pressure, different liquids will boil at different temperatures.

The **normal boiling point** (also called the **atmospheric boiling point** or the **atmospheric pressure boiling point**) of a liquid is the special case in which the vapor pressure of the liquid equals the defined atmospheric pressure at sea level, one atmosphere.[4][5] At that temperature, the vapor pressure of the liquid becomes sufficient to overcome atmospheric pressure and allow bubbles of vapor to form inside the bulk of the liquid. The **standard boiling point** has been defined by IUPAC since 1982 as the temperature at which boiling occurs under a pressure of one bar.[6]

The heat of vaporization is the energy required to transform a given quantity (a mol, kg, pound, etc.) of a substance from a liquid into a gas at a given pressure (often atmospheric pressure).

Liquids may change to a vapor at temperatures below their boiling points through the process of evaporation. Evaporation is a surface phenomenon in which molecules located near the liquid's edge, not contained by enough liquid pressure on that side, escape into the surroundings as vapor. On the other hand, boiling is a process in which molecules anywhere in the liquid escape, resulting in the formation of vapor

After pre-training RoBERTa-large, we apply fine turning on the model. We use 50 epochs, batch size 8, seed 42 and set learning rate to be 1e-5.
Same as RoBERTa-base, training data is divided into 5 folds for RoBERTa-large. RMSE of model finetuned on each fold is shown as following:

RMSE	RoBERTa-large
Augmented Pretrain + finetune fold1	0.484
Augmented Pretrain + finetune fold2	0.462
Augmented Pretrain + finetune fold3	0.466
Augmented Pretrain + finetune fold4	0.508
Augmented Pretrain + finetune fold5	0.447
Average scores	0.474

Table 8 the RMSE of RoBERTa-large with pre-training

It can be seen the distribution from the labels and the predicted values on training dataset are coincide in range [-2.6, -2.0].

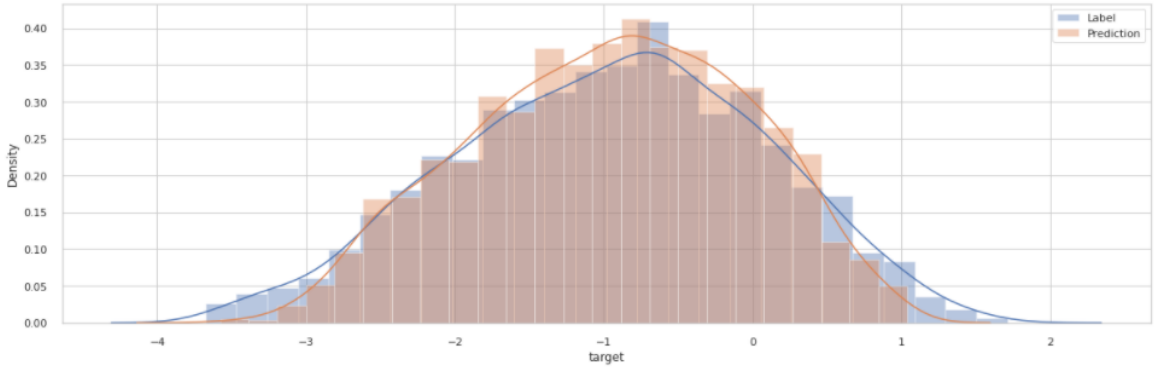


Figure 7 the distribution from the labels and the predictions of RoBERTa-large without pre-training

3.4 RoBERTa with SVM

Deutsch, Jasbi, and Shieber (2020) [11] managed to use the predictions of BERT as features in a SVM model in text readability task. And the BERT-based model outperforms the second-best BiLSTM by about 8%. Additional to predictions of BERT, a set of syntactic and lexico-semantic features is also utilized. However, the BERT-based model did not improve with the additional syntactic and lexico-semantic features. The reason why a set of syntactic and lexico-semantic features did not work well may be that BERT has already learned all the information presented by syntactic and lexico-semantic features.

This study inspires us to use the predictions of BERT-based model as features feeding into regression model like SVM in our task. We decide to abandon additional syntactic and lexico-semantic features because it does not help improve the performance and focus on the structure of model and corpora.

We use the structure in 3.2.1, a RoBERTa coming with an Attention-head layer and regression layer. The Attention-head layer is to gain a stronger feature embedding from the output of RoBERTa-large model. But the regression layer is replaced by SVM or other regression models. The hidden sizes of Attention-head is1024, differentiate from the Attention-head using during finetune (512 hidden sizes). The output of Attention-head layer is the embeddings of excerpts. It is also the input of regression model. The process of getting embedding can be described as followed.

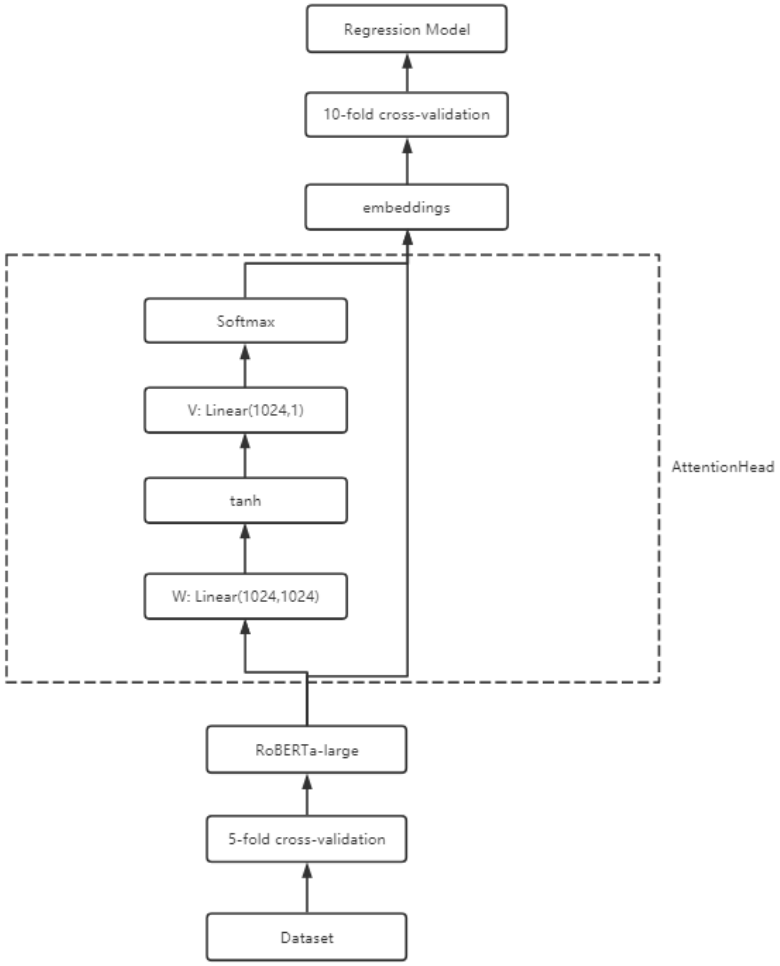


Figure 8 the Process of Getting Embeddings

Thus, each text in training data is represented by a vector with 1024 dimensions, which is the embedding of the text. The embeddings are further fed into a SVM classifier to get the final prediction. Because our task is a regression task, we construct a SVR model using the Scikit-Learn library (Pedregosa et al.,2011) [12]. The parameter optimization was using the suggestion by Deutsch, Jasbi, and Shieber (2020) [11]. SVR is the use of SVM in regression and initially proposed by Drucker et al., (1997) [13]. The purpose of SVR is to reduce the error by determining the hyperplane and minimizing the range between the predicted and the observed values (Singh 2020) [14]. The structure of SVR is shown as followed. The points in the figure represents the observed values. The green points are the values on margin and red points are outliers.

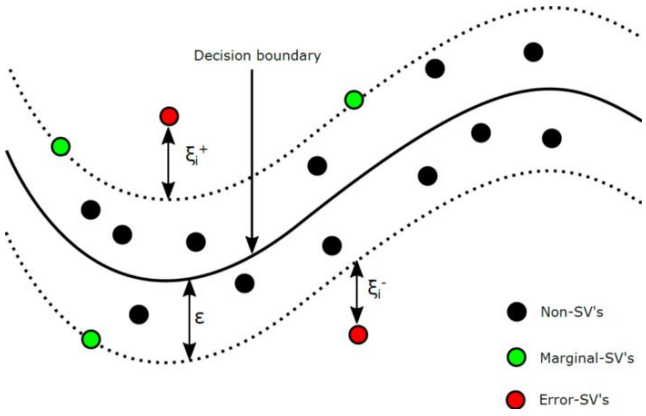


Figure 9 the structure of SVR

Considering feeding the embedding generated by BERT-based model may cause overfitting, we decide to use 10-fold cross-validation when feeding each fold's embeddings generated by RoBERTa-large into SVR. The mean RMSE is 0.39, which is the best CV in our task. It means feeding the embeddings of BERT-based model into a regression model is a feasible method. We then tested a variety of regression models, including linear regression, Ridge, Lasso, RandomForest, Adaboost, etc. We compared these models with 10-fold cross-validation. The parameters and results are shown as followed.

model	parameters
Linear regression	auto
Ridge	alpha=10
Lasso	alpha=0.1, normalize=True
SVR	kernel='rbf', gamma='auto'
RandomForest	n_estimators=100
Adaboost	n_estimators=300

Table 9 the parameters of regression models

Since we use 5-fold cross-validation when training RoBERTa-large, we will have 5 different RoBERTa-large models to generate 5 different sets of embeddings with the shape (2834,1024). For each set of embeddings, we further feed them into regression model using 10-fold cross-validation

The figure below shows the local mean RMSE of each fold of embeddings while feeding them to different regression model. As we can see from the figure, the performance of SVR is the best. Thus, we decide to use it for further process.

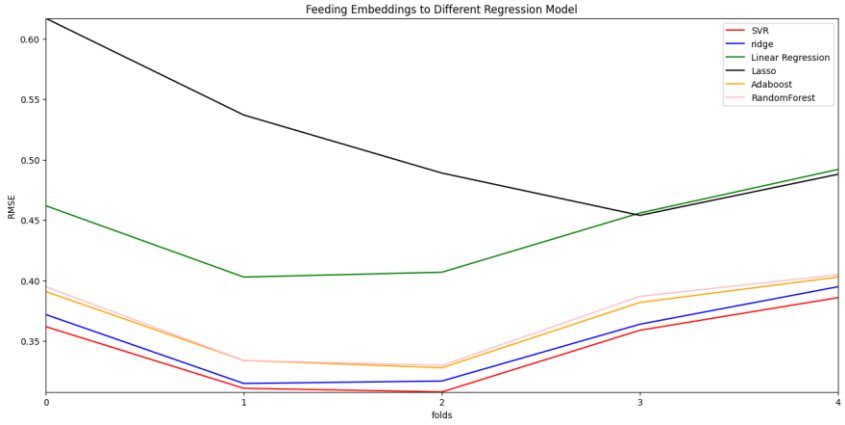


Figure 10 Feeding Embeddings to Different Regression Model

3.5 Ensemble

3.5.1 Stacking ensemble

Stacking builds its models using different learning algorithms. And then a combiner algorithm is trained to make the ultimate predictions using the predictions generated by the base algorithms. This combiner can be any ensemble technique.错误!未找到引用源。 Since we believe the stacking would be the most suitable method for us to improve the prediction accuracy, we employed a scheme formed by two basic models and a meta-model. At the base level, we chose the RoBERTa-base and RoBERTa-large model with the highest cv score. At the meta-level, the Ridge Regression Model had been used to combine the predictions produced by the base level. The employed scheme is graphically shown in Figure 10.

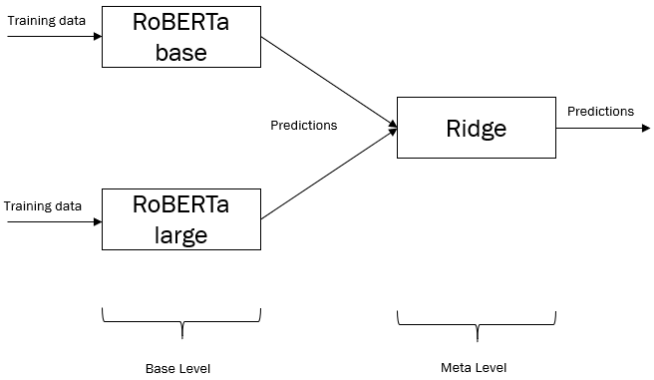


Figure 10. A graphical representation of the stacking ensemble method used in this project

To avoid overfitting, the 5-fold will be used in the stacking ensemble. The distribution of training dataset for meta model is shown in Figure 11, and the mean of 5-fold target is shown in table 9.

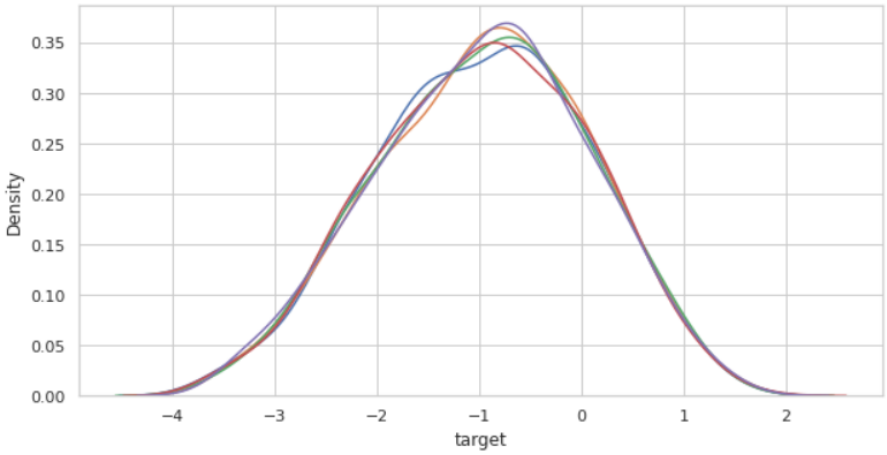


Figure 11. The distribution of training dataset for meta model

k-fold	The mean of target
1	0.961168
2	0.957628
3	0.963653
4	0.956987
5	0.957154

Table 9. The mean of 5-fold target

3.5.2 Blending fold 4

After researching the performance of each fold in all models we trained, we found that they all did not perform well in fold 4. In this project, we have trained a lot of models. The best RoBERTa-base model will be represented as model 4 while the best RoBERTa-large one will be represented as model 19. The RMSE

score of fold 4 in all basic models is shown in table 10.

Model	RMSE Score
RoBERTa-base (model 4)	0.505
RoBERTa-large (model 19)	0.508

Table 10. The RMSE Score of fold 4

The model trained with the training content of fold 4 may be beneficial to the overall effect. In other words, the model in fold 4 is trained with the training content of folds 1,2,3,5. But it does not perform well in fold 4. We have tried to change the distribution of training data in the part of fine-tuning. But it still cannot improve the accuracy. To improve the accuracy of this part, we tried to blend with the model of SVM + RoBERTa-large in fold 4. As we know that SVM + RoBERTa-large model fits well with the training data. However, in this case, we try to use the blending of 0.7 * 0.3 to ensure that our predictions are slightly better but not over-fitting. In the following, the details of the scheme will be graphically shown.

The graph of training data for meta model is shown in figure 12.

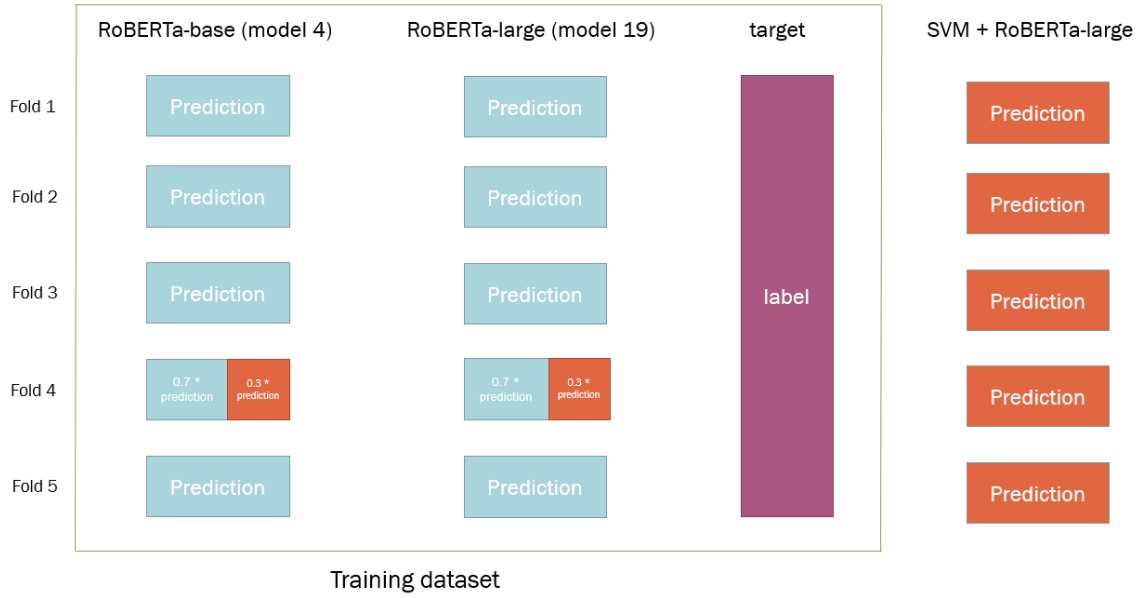


Figure 12 training dataset for meta-model

The figure of test data for meta model is shown in figure 13.

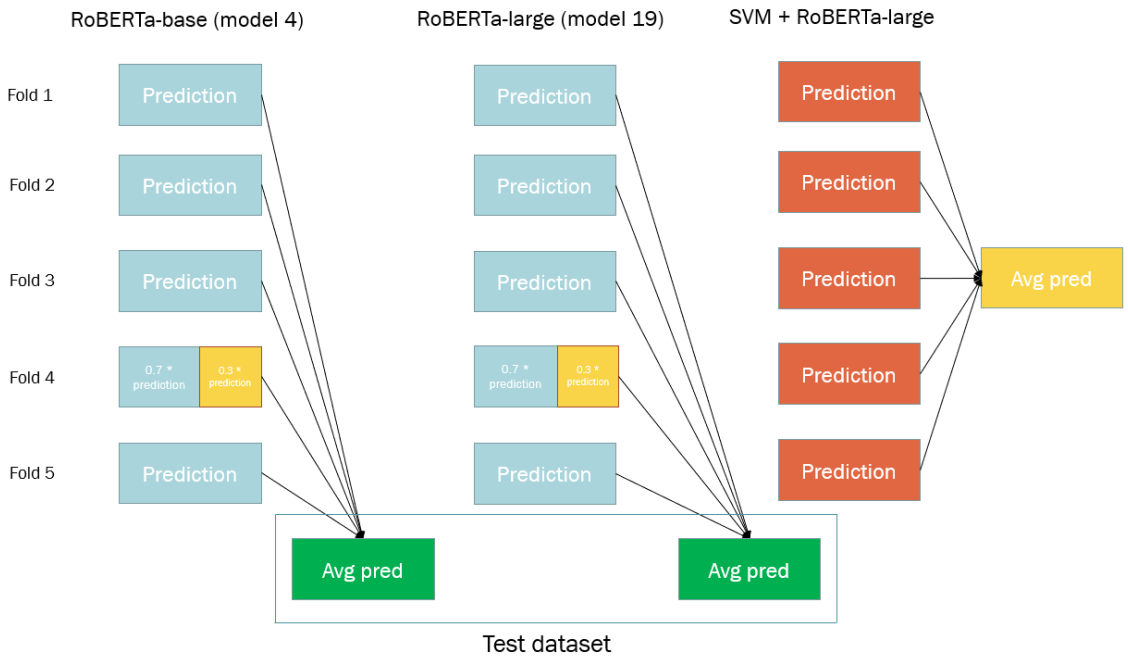


Figure 13 test dataset for meta-model

The overall scheme is shown in figure 14.

Figure 14 the scheme of stacking ensemble

3.5.3 Ablation study

To investigate the behavior of the blending fold 4, we set the stacking model 4 and model 19 as the baseline and conducted several ablation studies.

First, we will separately show the effect of this method on the individual model without stacking. To do this, each model's average prediction just like figure 13 shown will be evaluated.

Next, this method will be used in one of the models in the stacking ensemble, and the other will remain unchanged. For example, model 4 implements the blending fold 4 while model 19 remains the origin structure.

Finally, we will compare the performance between stacking ensemble without blending fold 4 and stacking ensemble with blending fold 4.

4. Experimental Setup

4.1 Dataset

4.1.1 Competition Dataset

The competition dataset (https://www.kaggle.com/c/commonlitreadabilityprize/data) is provided by Kaggle, the competition holder. It contains excerpts from several time periods and a wide range of reading ease scores. The size of competition dataset is 2.79MB and contain the total of 2834 data.

4.1.2 Augmented Dataset

We use Project Gutenberg (https://www.kaggle.com/nltkdata/gutenberg) as augmented dataset along with competition dataset for pretraining. we first search the source article of excerpts in competition dataset from Gutenberg and managed to cut out the excerpts before and after the original excerpt in competition dataset. We cut off each augmented data to ensure the length is less than 510 tokens because the maximum length of BERT input should be limited to 510 tokens(includes [CLS] and [SEP]). We finally sorted out 802 rows of augmented data for pretraining. The num of tokens used for pretraining before adding augmented data is 490231. After adding augmented data, the numbers of tokens in total become 1567640.

4.2 Evaluation Metrics

In this task, the result of predict model will be evaluated on the root mean squared error. RMSE is defined as:

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2},$$

in which y_i is target and $\hat{y_i}$ is our model predicted result.

4.3 Terms

CV means the local cross validation score.
Private score means it is calculated with approximately 70% of the test data in Kaggle.
Public score means it is calculated with approximately 30% of the test data in Kaggle.

5. Results

5.1 Machine Learning

model	cv	private	public	run time
Linear regression	0.667	0.667	0.652	1.52''
Ridge	**0.637**	**0.649**	**0.624**	1.07''
Lasso	1.045	1.028	1.048	**0.83''**
SVR	0.958	0.934	0.927	41.54''
RandomForest	0.714	0.711	0.694	8'7''
Adaboost	0.703	0.697	0.675	2'46''

Table 11 the result of Machine Learning methods

5.2 RoBERTa Models

	model	cv	private	public
Single Model	RoBERTa-base without pre-train	0.477	0.470	0.467
	RoBERTa-base pre-trained with competition dataset (model 4)	0.475	0.470	0.467
	RoBERTa-large without pre-train	0.481	0.465	0.465
	RoBERTa-large pre-trained with augmented dataset (model 19)	0.474	0.463	0.465
	RoBERTa-large with SVM	**0.346**	0.472	0.468
	RoBERTa-large with Ridge	0.352	0.472	0.469
	RoBERTa-large with Adaboost	0.368	0.481	0.476
	RoBERTa-large with RandomForest	0.370	0.484	0.477
Single Model with blending	RoBERTa-base (model 4) + RoBERTa-large with SVM	0.464	0.468	0.466
	RoBERTa-large (model 19) + RoBERTa-large with SVM	0.460	0.463	0.465
Stacking	RoBERTa-base (model 4) + RoBERTa-large (model 19)	0.461	**0.457**	0.458
	RoBERTa-base (model 4 blending) + RoBERTa-large (model 19)	0.458	0.458	0.458
	RoBERTa-base (model 4) + RoBERTa-large (model 19 blending)	0.456	**0.457**	0.458
	RoBERTa-base (model 4 blending) + RoBERTa-large (model 19 blending)	0.450	**0.457**	**0.457**

Table 12 the result of RoBERTa models

6. Discussion

6.1 Machine Learning

It can be seen from the results of the machine learning models that Ridge and linear regression are more effective, which achieved the score of 0.649 and 0.667 in private score respectively. Overall, compared to the RoBERTa models, the machine learning models run much faster. But the prediction accuracy of machine learning models is poor. Even the Ridge with the best prediction effect has a huge gap with the RoBERTa models. As analysed in 3.1, the training data set of the project is relatively small. So, it is difficult to train the model to the desired effect. In addition, it is difficult to extract more text information due to the limited feature extraction methods. Therefore, trying to find more training data sets or trying better feature extraction methods is the way to enhance the effect of the machine learning models.

6.2 RoBERTa-base

The result of RoBERTa-base without pre-train and RoBERTa-base pre-train with competition dataset are both 0.470 on private 0.467 on public. They do not have obvious difference, which means pretraining on competition dataset does not obviously improve the performance of RoBERTa-base.

6.3 RoBERTa-large

But when comparing the result of RoBERTa-large without pre-train and RoBERTa-large pre-trained with aux dataset, it is noticed that the cv, private, public score of pre-trained RoBERTa-large are much higher

than non-pretrained RoBERTa-large. Pre-trained RoBERTa-large has private score 0.463 and public score 0.465. Therefore, pre-training on the Augmented Dataset is an effective way to improve the performance of RoBERTa-large.

6.4 SVM + RoBERTa-large

In the process of feeding the embeddings generated by BERT-based model into the regression model, SVR, Ridge, Adaboost, and RandomForest perform much better than other regression models. Among them, SVR has the best score on both CV and public board. Feeding embeddings into the regression model can learn the information that BERT-based model cannot. And it also addresses the issue that traditional feature extraction cannot fully extract the features from training data. However, this method also has a fatal problem of overfitting. The local CV score is 0.346, which is much better than other single models. But it only gets 0.472 on the private score.

6.5 Ablation study

On the one hand, the private score of model 4 using blending fold 4 decreases from 0.470 to 0.468 while the public score of that reduces from 0.467 to 0.466. On the other hand, the private score of model 19 using blending fold 4 is 0.463 while the public score of that is 0.465. It is the same as the one without using blending fold 4.

The public score and private score of the baseline are 0.457, 0.458. When model 4 implements blending fold 4 while model 19 remains the origin structure, the public score increases from 0.457 to 0.458 while the private score remains the same. When model 19 uses blending fold 4 while model 4 maintains the same, the public score and the private score are the same as the baseline.
When we use the scheme in figure 5, the public score decreases from 0.458 to 0.457 while the private score is still the same as the baseline.

Consequently, blending fold 4 could improve the accuracy of single model without stacking ensemble. However, if this method is implemented in one of the models in the stacking ensemble, the performance cannot be improved. In comparison, if the model 4 and model 19 are all implemented blending fold 4, the public score can increase 0.01. For this employed scheme, this method can indeed slightly improve the effect, but it is not obvious.

7. Conclusions

In this report, we explore serval algorithms to rate the complexity of reading passages, ordered roughly by time sequence or public/private score. During our first trial, we used tradition feature extraction and fed the features to machine learning regression models. The highest score provided by tradition machine learning method is around 0.67. However, BERT-based models performed much better in this task, even the models are not pretrained and finetuned. Thus, machine learning methods are abandoned, and BERT-based models are onboard.

During our second phase, we tried to find a best-perform BERT-based model. We used more corpus to pretrain, changed parameters and structures while finetuning and combining BERT-based models with machine learning methods. After trying different BERT-based models, we found that RoBERTa is much suitable for our task. RoBERTa-large performs better than RoBERTa-base. And pretraining RoBERTa with more corpora can slightly improve the score. We also found that feeding the embedding generated by BERT-based models into regression models is also an effective way. The limit score of a single BERT-based model is 0.467 on public board and 0.472 on private board.

At last, stacking is used as an ensemble learning method in our task. We ensembled different BERT-based models to find the best model. We blended one RoBERTa-base model and one RoBERTa-large model. Then we stacked blended models with a meta-model. The highest score (0.457) in our experiments is achieved by this model.

For future development, some method can be used to improve the performance of model. First of all, we can try to choose other models which may have better performance, including Deberta-base, Deberta-large, sentence-transformers/LaBSE, Xlnet.Secondly. we can external data selection for pretraining or training, which means use sentence-transformers generate sentence embeddings, create retrieve snippets and choose highest cosine similarity to the original excerpt. Thirdly, use models which adopts a supervision method of simultaneous training on pseudo labeled and unlabeled data. At last, multi-level stacking may perform better than original stacking. This stacking separate models into levels. For example, on level1 place all the low-score models, on level2 place pseudo label models, on level3 place linear models.

8. References

[1]. Edgar Dale and Jeanne S. Chall. 1949. The concept of readability. Elementary English, 26(1):19–26.
[2]. Kincaid, J.P., Fishburne Jr, R.P., Rogers, R.L., Chissom, B.S.: Derivation of new readability formulas for navy enlisted personnel (1975).
[3]. Todirascu, Amalia, Thomas François, Delphine Bernhard, Núria Gala, and Anne-Laure Ligozat. 2016. Are cohesive features relevant for text readability evaluation? In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 987–997, Osaka, Japan.
[4]. Collins-Thompson, Kevyn. 2014. Computational assessment of text readability: A survey of current and future research. ITL-International Journal of Applied Linguistics, 165(2):97 135.
[5]. Xia, M., E. Kochmar & T. Briscoe (2016). Text readability assessment for second language learners. In Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications. pp. 12–22.
[6]. Matej Martinc, Senja Pollak, and Marko RobnikSikonja. 2019. ˇ Supervised and unsupervised neural approaches to text readability. Computing Research Repository, arXiv:1503.06733. Version 2.
[7]. Changping Meng, Muhao Chen, Jie Mao, and Jennifer Neville. 2020. Readnet: A hierarchical transformer framework for web article readability analysis. In European Conference on Information Retrieval, pages 33–49. Springer.
[8]. RoBERTa: A Robustly Optimized BERT Pretraining Approach
[9]. REVISITING FEW-SAMPLE BERT FINE-TUNING
[10]. Martinc, M., Pollak, S. and Robnik-Šikonja, M., 2021. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, *47*(1), pp.141-179.
[11]. Deutsch, T., Jasbi, M. and Shieber, S., 2020. Linguistic features for readability assessment. *arXiv preprint arXiv:2006.00377*.
[12]. Fabian Pedregosa, Gael Varoquaux, Alexandre Gram- ¨ fort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. ´ Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825-2830.
[13]. Drucker H, Burges C J C, Kaufman L, et al. Support vector regression machines[J]. Advances in neural information processing systems, 1997, 9: 155-161.
[14]. Abhilash Singh. Support Vector Regression (SVR) Model: A Regression-Based Machine Learning Approach[EB/OL]. [2021/8/10]. https://medium.com/analytics-vidhya/support-vector-regression-svr-model-a-regression-based-machine-learning-approach-f4641670c5bb.
[15]. F. Divina, A. Gilson, F. Goméz-Vela, M. García Torres, J.F. Torres, stacking ensemble learning for short-term electricity consumption forecasting, Energies 11 (4) (2018) http://dx.doi.org/10.3390/en11040949

9. Appendix

Model 4: https://www.kaggle.com/andretugan/commonlit-roberta-0467
Model 19: https://www.kaggle.com/joechan619/roberta-large-5fold-aux
SVM + RoBERTa-large Model: https://www.kaggle.com/joechan619/svm-robertalarge