A Survey of Solving Out-of-Domain Generalization Problems in Automated Holistic Essay Scoring

Qin Ma*, Yuquan Zhou*, Bo Li*
Fudan University
School of Data Science
{21307110024, 21307100050, 21307110183}@m.fudan.edu.cn

Abstract

Automated Essay Scoring (AES) technology is becoming increasingly important in today's educational landscape. Natural language processing based on neural networks has the potential to advance this technology. However, the generalization ability of neural network models often requires a large amount of high-quality, labeled data. In situations where this condition is not met, how to enhance the model's generalization capability becomes a key issue that hinders AES performance. Our team has conducted an in-depth exploration of this issue. By combining DeBERTa and feature engineering, we proposed **GenScorer** under an ensemble learning framework. We validated our algorithm on the **Kaggle competition AES 2.0**, where **GenScorer** significantly improved the performance of the baseline.

1 Introduction

Automated essay scoring (AES), the task of employing computer technology to score written text, is one of the most important educational applications of natural language processing (NLP). The vast majority of work on AES has focused on holistic scoring, which summarizes the quality of an essay with a single score. This involves calculating and analyzing various linguistic features such as grammar, spelling, syntactic structure, vocabulary, coherence, and logicality(Ke Z, 2019) .Holistic scoring technologies are commercially valuable: being able to automate the scoring of the millions of essays written for standardized aptitude tests can save a lot of manual grading effort.

However, obtaining a large number of high-quality labeled papers is not easy, and the small size of the datasets can lead to models overfitting to the source domain but performing poorly across multiple different target domains. Moreover, data from target domains is often difficult to obtain, making it challenging to simply use domain adaptation techniques to improve the model. Therefore, our research focuses more on the model's generalization ability on unknown samples.

Learning Agency Lab - Automated Essay Scoring 2.0 is a Kaggle competition that focuses on the accuracy of holistic scores. The competition does not provide a test set, making the model's generalization ability crucial. We choose this competition to validate our algorithm.

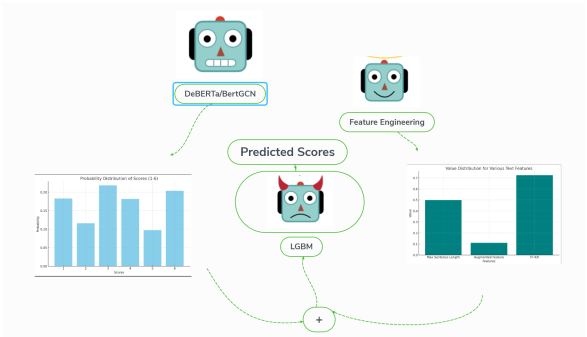

Figure 1: Illustration of GenScorer Architecture

2 Related Work

AES has developed rapidly and has achieved many research results worldwide.We will primarily review previous works in AES from three aspects: feature engineering-based machine learning methods, deep learning-based methods, and hybrid models.

2.1 Feature engineering-based machine learning methods

Feature engineering-based machine learning methods for Automated Essay Scoring involve the manual construction of features based on linguistic, grammatical, and syntactical rules. These methods rely on the expertise of linguists to identify relevant features that can effectively represent the quality and content of essays. Traditional machine learning algorithms such as Logistic Regression (LR)

and Support Vector Machines (SVM) are then used to train models on these features and score essays. Amorim et al.(Amorim E, 2018) were among the first to propose modeling approaches for AES and utilized features like grammatical errors and syntactical information, employing an LR model for scoring essays.Valenti et al. (Valenti S, 2003) improved traditional vocabulary features and applied them to the automatic scoring of essays. Haberman et al. (Haberman SJ, 2010) extracted features from three levels: structure, syntax, and vocabulary, and constructed an automatic essay scoring model based on an LR model and discourse structure. Hussein et al. (M.A. Hussein, 2019) developed an AES system utilizing various handcrafted features. Additionally, PEG (Project Essay Grade) (Dikli, 2006), one of the earliest AES systems, scored essays based on shallow semantic features such as writing structure. Burstein et al. (Ifenthaler, 2022a) created features from content , structure and organization, employing a ranking method for essay scoring. Darwish et al. (Ifenthaler, 2022b) introduced two semantic features, namely essay consistency and coherence, to evaluate essay content. Chen et al. (S.M. Darwish, 2020) proposed an unsupervised AES system.

2.2 Deep learning-based methods

Recently, deep learning methods based on neural networks have been rapidly developing and achieving significant research outcomes in the field of automated essay scoring. Taghipour and Ng (Taghipour and Ng, 2016) create models based on RNN have been proposed .He et al. (He Y, 2023) used a hybrid neural network model to identify beautiful sentences in essays. Efendi et al. (Efendi T, 2022) utilized ConvNet to extract features related to sentence fluency and integrated them with topic features for essay scoring. Cai et al. (C, 2019) applied ConvNet, LSTM, and other neural network methods to automatically extract essay features. Jin et al. (Jin C, 2018) employed a dual-layer Bi-LSTM network for automated essay scoring. Jin et al. (Zhang Y, 2021) also introduced a two-stage neural network model, which performed well in evaluating essays irrelevant to the topic. Parekh et al. (Kumar Y, 2023) proposed the SkipFlow model, which better simulates semantic relationships in long texts. Rodriguez et al. (Morris W, 2023) applied BERT and XLNet models in the AES field and achieved good performance. Wang et al. (Ma C, 2024) proposed the Multi-level Feature Fusion model, which captures lexical, sentence, and chapter-level features through various sub-models and integrates these features using ConvNet and BiLSTM networks for essay scoring. Gupta et al. (K, 2023) developed an AUG method to enhance generalization performance by periodically enhancing essays with their topics, utilizing a series of models including BERT, RoBERTa, ALBERT, DistilBERT, and XLM-RoBERTa for data augmentation and prediction tasks.

2.3 Hybrid models

Many studies have shown that combining neural network models with manual features can effectively improve the performance of automatic essay scoring. Nadeem et al. (Yamaura M, 2023) proposed features such as topic relevance and integrated them with features based on word embedding representation, proposing a multi model fusion method for automatic essay scoring. Liu et al. (Lee J, 2024) used a Graph Neural Network (GNN) model and integrated fluency and text matching features into the neural network to extract essay features. Mayfield et al. (Mayfield E, 2020) fused character level n-gram features with word embedding features to extract semantic features, achieving good performance on the ASAP dataset. Liu et al. (Yang K, 2024) introduced a two-stage learning framework (TSLF) that uses neural network models to extract semantic, fluency, and correlation features, and automatically grades essays after integrating manual features. Morris et al. (Revathy JS, 2024) developed a complex deep learning based method that utilizes BERT to learn vocabulary representations and utilizes a bidirectional long short-term memory network (BiLSTM) with attention mechanism for scoring. Their method, BERT BiLSTM ATT, directly solves the underfitting problem through dynamic loss function enhancement, representing a hybrid model that combines attention mechanism with deep learning architecture. Hendre et al. (Hendre M, 2023) introduced a deep learning based automatic essay evaluation system, GSE Large, which utilizes advanced neural embedding methods such as Google Sentence Encoder (GSE), ELMo, and GloVe for semantic similarity calculation. The combination of traditional text data representation methods such as TF-IDF and Jaccard exponent, as well as neural embedding, marks a hybrid approach that combines deep learning and feature engineering techniques for semantic anal-

ysis. Yao et al. (Yao L, 2023) employed a hybrid model that utilizes BERT and various NLP techniques for corpus processing and feature extraction, and combined traditional machine learning classifiers with neural network-based methods for essay scoring.

Previous studies have shown that both shallow features based on feature engineering and deep latent semantic features based on deep learning can effectively improve automatic essay scoring. So in this project, we used DeBERTa and LightGBM to predict the scores of essays.

3 Corpus and Tasks

3.1 Corpus

The competition dataset comprises about 24000 student-written argumentative essays. Each essay was scored on a scale of 1 to 6. The dataset is divided into two parts: training and testing. Competitors are only available to the training set (about 17000 essays). Our goal is to extract universal features applicable to essays on all topics from a limited training set and predict the scores of testing set.

Persuade 2.0 contains 25000 essays, which are also argumentative essays, and each essay is scored on a scale of 1 to 6. We use this dataset to expand the training set, but the results are not as good as expected because of the domain gap between Persuade 2.0 and the competition testing set, so we do not use this dataset in the final model.

3.2 Tasks

Predictions are evaluated based on the quadratic weighted kappa, which measures the agreement between two outcomes. This metric typically varies from 0 (random agreement) to 1 (complete agreement). In the event that there is less agreement than expected by chance, the metric may go below 0.

The quadratic weighted kappa is calculated as follows. First, an N x N histogram matrix O is constructed, such that $O_{i,j}$ corresponds to the score of essay_id i (actual) that received a predicted value j. An N-by-N matrix of weights, w, is calculated based on the difference between actual and predicted values:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

An N-by-N histogram matrix of expected outcomes, E, is calculated assuming that there is no correlation between values. This is calculated as the outer product between the actual histogram vector of outcomes and the predicted histogram vector, normalized such that E and O have the same sum.

From these three matrices, the quadratic weighted kappa is calculated as:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

The closer the model's scores are to the actual scores, the closer the value of κ will be to 1. Our goal is to achieve the highest possible κ score.

4 GenScorer: A Generalized Scoring Model

We now introduce our framework of GenScorer and how it enables out-of-domain generalization in automated holistic essay scoring.

4.1 Baseline

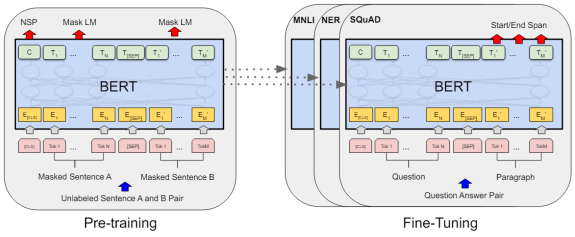

Figure 2: BERT structure: The left image represents the pre-training process, while the right image shows the fine-tuning process for specific tasks.The picture is from https://arxiv.org/pdf/1810.04805

We use DeBERTa as the backbone model for our baseline.

DeBERTa(Decoding-enhanced BERT with disentangled attention) improves BERT(Bidirectional Encoder Representation from Transformers) with three novel components: disentangled attention, an enhanced mask decoder and scale invariant fine-tuning. First, we will introduce the model architecture of DeBERTa.

Disentangled Attention: A Dual-Vector Approach for Content and Position Embeddings

In BERT, each token is represented by a single vector, which is the sum of the word (content) embedding and the position embedding. In contrast, DeBERTa decouples the token embedding and uses two vectors to represent a token: content and relative position. For token i, DeBERTa represents it as $\{H_i\}$ (content) and $\{P_{i|j}\}$ (relative position).

The attention score between token i and token j can be decomposed into four parts: content-to-content, content-to-position, position-to-content, and position-to-position:

$$\begin{aligned} A_{i,j} &= \{H_i, P_{i|j}\} \times \{H_j, P_{j|i}\}^T \\ &= H_i H_j^T + H_i P_{j|i}^T + P_{i|j} H_j^T + P_{i|j} P_{j|i}^T \end{aligned}$$

Due to the use of relative positional encoding, $P_{i|j} P_{j|i}^\top$ is unimportant and omit it. The specific implementation is as follows. Let $k = 512$ be the possible maximum relative distance, $\delta(i,j) \in [0, 2k)$ be the relative distance between token i, j:

$$\delta(i,j) = \begin{cases} 0 & \text{for } i - j \leq -k \\ 2k-1 & \text{for } i - j \geq k \\ i - j + k & \text{others} \end{cases}$$

The above relative distance can be represented using an embedding matrix $P \in \mathbb{R}^{2k \times d}$ (shared across all layers). Disentangled self-attention with relative position bias can be expressed as follows:

$$\begin{aligned} Q_c = HW_{q,c}, \quad K_c = HW_{k,c} \\ V_c = HW_{v,c}, \quad Q_r = PW_{q,r} \\ K_r = PW_{k,r} \\ \tilde{A}_{ij} = Q_i^c K_j^{cT} + Q_i^c K_{\delta(i,j)}^{rT} + K_j^c Q_{\delta(j,i)}^{rT} \\ H_o = \text{softmax}\left(\frac{\tilde{A}}{\sqrt{3d}}\right) V_c \end{aligned}$$

Among them, $Q_i^c K_j^{cT}$ is content to content, $Q_i^c K_{\delta(i,j)}^{rT}$ is content to position, $K_j^c Q_{\delta(j,i)}^{rT}$ is position to content. $H \in \mathbb{R}^{N \times d}$ and $P \in \mathbb{R}^{2k \times d}$ are the content vectors and relative position vectors respectively, where N is the number of tokens. $W_{q,c}, W_{k,c}, W_{v,c} \in \mathbb{R}^{d \times d}$ are the projected content vectors. $Q_c, K_c, V_c \in \mathbb{R}^{N \times d}$ are the corresponding projection matrices. $W_{q,r}, W_{k,r} \in \mathbb{R}^{d \times d}$ are the projected relative position vectors. $Q_r, K_r \in \mathbb{R}^{N \times d}$ are the corresponding projection matrices. $A_{i,j}$ is the element of the attention matrix $\tilde{A}$, representing the attention score from token i to token j. Q_i^c is the i-th row of Q_c. K_j^c is the j-th row of K_c. $K_{\delta(i,j)}^r$ is the $\delta(i,j)$-th row of K_r with regarding to the relative distance $\delta(i,j)$. $Q_{\delta(j,i)}^r$ is the $\delta(j,i)$-th row of Q_r with regarding to the relative distance $\delta(j,i)$.

Algorithm 1 Disentangled Attention

Input: Hidden state H, relative distance embedding P, relative distance matrix δ. Content projection matrices $W_{k,c}$, $W_{q,c}$, $W_{v,c}$, position projection matrices $W_{k,r}$, $W_{q,r}$.
Output: Output H_o
$K_c = HW_{k,c}, Q_c = HW_{q,c}, V_c = HW_{v,c}$
$K_r = PW_{k,r}, Q_r = PW_{q,r}, A_{c\to c} = Q_c K_c^T$
for $i = 0$ to $N-1$ **do**
    $A_{c\to p}[i,:] = Q_c[i,:] K_r^T$
for $i = 0$ to $N-1$ **do**
    for $j = 0$ to $N-1$ **do**
        $A_{c\to p}[i,j] = A_{c\to p}[i,j] + \delta(p_{i,j})$
for $j = 0$ to $N-1$ **do**
    $A_{p\to c}[j,:] = K_c[j,:] Q_r^T$
for $j = 0$ to $N-1$ **do**
    for $i = 0$ to $N-1$ **do**
        $A_{p\to c}[j,i] = A_{p\to c}[\delta(j,i), j]$
$\hat{A} = A_{c\to c} + A_{c\to p} + A_{p\to c}$
$H_o = \text{softmax}\left(\frac{\hat{A}}{\sqrt{3d}}\right) V_c$
return H_o

Algorithm 1 describes the disentangled attention mechanism in DeBERTa.

Enhanced Mask Decoder Accounts for Absolute Word Positions

To compensate for absolute position information, the BERT model incorporates absolute positions in the input layer. In contrast, DeBERTa adds absolute word position embeddings directly before the softmax when performing masked token prediction, which uses the softmax function to determine the probability of each word for the masked token. DeBERTa captures relative positions throughout all Transformer layers, using absolute positions only as complementary information when decoding masked words.Previous experiments indicate that Enhanced Mask Decoder performs better.

Scale Invariant Fine-Tuning

DeBERTa uses a new adversarial training algorithm called Scale-invariant-Fine-Tuning (SiFT) during fine-tuning. In NLP tasks, adversarial training is typically applied to word embeddings. However, the norms of word embeddings vary for different tokens, and the variance increases with the size of the model parameters, leading to instability in the training process. To address this, SiFT

first normalizes the word embeddings into probability vectors, and then adds perturbations to the normalized word embeddings.

Sequential Classification

The model uses the last hidden state of the `[CLS]` token as the input to the classification layer. The classification layer is a linear layer with a softmax activation function. The output of the classification layer is a probability distribution over the possible scores range from 1 to 6.

We use 2 Nvidia 4090 GPUs to train the model. And get 0.768 on the public leaderboard.

Figure 3: baseline scores

4.2 BertGCN

Though Bert is pretty powerful, it fail to capture the graph structure that can represent the relationships and interactions between multiple documents and words. This limitation can hinder the performance of text classification tasks where understanding the inter-document relationships is crucial. In our AES task, we notice that most of the articles are argumentative ones. It is crucial for the model to be able to understand and capture potential relationship between words, sentences and paragraphs. Also, the organization or structure of the articles may hugely influence the score, instead of abundant vocabulary or certain styles of writing that may appear more important in poems, prose or novels.

While there are various other ways to investigate the strutural organization of an essay(including feature engineering), BertGCN provides a more generous solution(Lin et al., 2021). It combines the strengths of BERT with Graph Convolutional Networks. By leveraging BERT's contextual embeddings and GCN's capability to capture graph structures, BertGCN constructs a heterogeneous graph over the dataset. Documents are represented as nodes with BERT-generated embeddings, and words are represented as additional nodes. By connecting the nodes, we can gain a graphical structure of the corpus. Inspired by methods that are familiar in computer vision, convolution-like methods are applied to the corpus graph.

4.2.1 Model Structure

The BertGCN model can be devided into two parts. The first step is to derive a graph structure of the corpus, including node and edges(with or without weight). The second step is to feed the feature of the node into a GCN model. The output logits of the GCN will then be utilized for classification, and final probability result will be weighted averaged with Bert result.

Building Corpus Graph The method adopted to develop a graph of the corpus is simple. First, we add all documents as Doc nodes and words as Word nodes to the graph. The we'll have to calculate the weight of edges and feeature of nodes.

(i) Weights of edges

We then calculate the TF-IDF matrix of the corpus, and use the value in the matrix as the weight of edges between Doc nodes and Word node. Then we calculate the PPMI matrix, and use the weight as weights between Word node. This can be expressed as follow:

$$A_{i,j} = \begin{cases} PPMI(i,j), & i,j \text{ words}, i \neq j \\ TF-IDF(i,j), & i \text{ document}, j \text{ word} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

However, in our implementation, we adopted cosine similarity for the edges between the nodes. This should also work.

(ii) Feature of the nodes

Feature of Word node are initialized as zeros and updated by gradient. Feature of the Doc nodes, however, are generated using Bert. That is, in the forward process of, input_ids and mask_matrix will be passed to Bert model, and the output at <CLS> position will be extracted as feature for the document. The feature matrix of all nodes can be expressed as:

$$X = \begin{pmatrix} X_{\text{doc}} \\ 0 \end{pmatrix}_{(n_{\text{doc}}+n_{\text{word}})\times d} \tag{2}$$

GCN Stage Now that we have an initialization of the graph, we can apply GCN on it. To be simple, GCN is the process of taking one node's feature and it's neighbors' feature, passing it through neural networks, and derive a new vector which and be used for classification task. The process can be

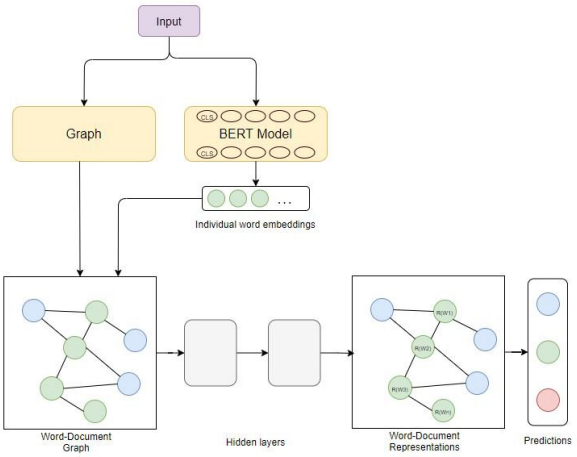

Figure 4: BertGCN Architecture
https://www.researchgate.net/figure/BERT-GCN-architecture_fig1_352374437

seen in picture4 To detail, the calculation of the hidden GCN layer can be express as:

$$L^{(i)} = \rho(\tilde{A}L^{(i-1)}W^{(i)}) \tag{3}$$

where ρ is an activate function, $\tilde{A}$ is the normalized adjacenccy matrix, and $L^{(i-1)}$ is the feature matrix(can stack several layers), whith $L^{(0)} = X$ is the input of GCN. The outputs will be fed to the softmax layer, i.e.:

$$(Z_{GCN}) = softmax(g(X, A)) \tag{4}$$

Note that Bert itself can provide a classification result.

$$(Z_{BERT}) = softmax(WX) \tag{5}$$

And we combined the result given by the two models to get:

$$Z = \lambda Z_{GCN} + (1 - \lambda)Z_{BERT} \tag{6}$$

4.2.2 Implementation

We implemented the model using PyTorch. The model ran through, bu unfortunately we don't have enough time to adjust the model and generate a result.

4.3 Stacked Ensemble Generalization

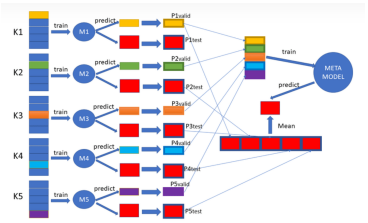

Figure 5: Stacked Generalization Architecture. Use the out-of-fold (OOF) data from each model in k-fold cross-validation, essentially the validation set, as features to construct the ensemble learning model, meta-model. And compare the average performance of the k base models with the performance of the meta-model.

Using the average of k-fold cross-validation to evaluate the performance of machine learning algorithms on a dataset is a common practice. This technique is most effective in assessing a model's ability to handle unknown samples, as it uses data that was not visible during training to evaluate the model.

The k out-of-fold (OOF) datasets from cross-validation can further be used to construct an ensemble learning model, also known as stacked generalization. Stacked generalization is trained using data that is unknown to the base models, thus it can correct the deficiencies of the base models in predicting new data.

In this task, we use the probability of score predictions from training samples using DeBERTa or BertGCN as one of the training features for ensemble learning. We employ the LGBM algorithm, which relies on features for training, making feature engineering particularly important.

4.3.1 Feature Engineering

In cases where the dataset is limited, neural network approaches tend to overfit to specific features of the source domain, and neural networks also suffer from poor interpretability and a black-box nature. Therefore, constructing general features for the target task is an important means of enhancing generalization performance.

Length-based features are one of the most important feature types for AES, as length is found to be highly positively correlated with the holistic score of an essay. These features encode the length of an essay in terms of the number of sentences, words, and/or characters in the essay. We have also

used statistical measures such as max, min, sum, mean, and kurtosis as features.

Since the dataset is composed of argumentative essays written by students, **argumentative features** are also important. These features include the number of argumentative words, the number of argumentative sentences, and the number of argumentative paragraphs in an essay. The argumentative structure of an essay is also a key factor in determining its quality. For instance, an essay typically has a major claim, which encodes the stance of the author w.r.t. the essay's topic. The major claim is supported or attacked by one or more claims (controversial statements that should not be readily accepted by the reader without further evidences), each of which is in turn supported or attacked by one or more premises (evidences for the corresponding claim).

Embeddings, which can be seen as a variant of n-gram features, are arguably a better representation of the semantics of a word/phrase than word n-grams.**TF-IDF** is a common method for generating embeddings. The importance of a word increases proportionally with its frequency of occurrence in a document, but this is offset by its frequency of appearance in the corpus, which helps to adjust the effect of words that typically appear more frequently.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Where $\text{TF}(t, d)$ is the frequency of term t in document d, and $\text{IDF}(t)$ is the inverse document frequency of term t.

We use above features to train the LGBM model. However, not all features are useful, thus **feature selection** is necessary. We choose features which variance is greater than 0.01 as the input of LGBM.

4.3.2 LGBM

Traditional GBDT needs to traverse the entire training data multiple times in each iteration. Loading the entire training data into memory limits the size of the training data; if the data is not loaded into memory, repeatedly reading and writing the training data consumes a significant amount of time. Especially when dealing with industrial-scale massive data, the standard GBDT algorithm cannot meet the requirements.

To address this issue, the LightGBM model introduces two new techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

Histogram-Based Algorithm

The greatest computational cost in training Gradient Boosting Decision Trees (GBDT) lies in finding the best split points. One of the most popular algorithms for finding split points is the pre-sorting algorithm, which traverses all possible split points in pre-sorted feature values. This method is straightforward and can find the optimal split points, but it is very time-consuming and memory-intensive. Another common algorithm for finding the best split points is the histogram-based algorithm. The histogram-based algorithm discretizes continuous feature values into bins and uses these bins to construct feature histograms during training.

LightGBM has further accelerated the histogram-based algorithm. The histogram of a leaf can be obtained by subtracting the histogram of its sibling from the histogram of its parent node, which can double the speed. Normally, constructing a histogram requires traversing all the data on that leaf, but histogram subtraction only requires traversing the k bins of the histogram. In the actual process of building trees, LightGBM can first compute the histograms of smaller leaf nodes and then use histogram subtraction to obtain the histograms of larger leaf nodes. This way, the histogram of the sibling leaf can be obtained at a very minimal cost.

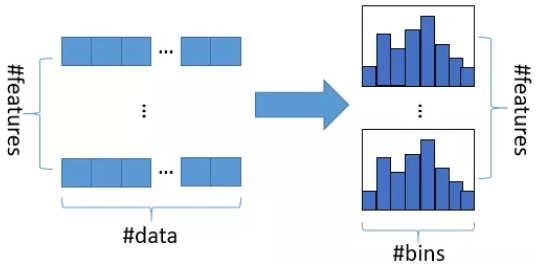

Figure 6: Histogram algorithm: LightGBM based on https://cloud.tencent.com/developer/article/1758058

Leaf-wise Algorithm with Depth Limitation

In addition to the Histogram algorithm, the LightGBM model has optimized the tree growth strategy. Instead of using the level-wise tree growth strategy employed by most GBDT tools, LightGBM uses a leaf-wise growth algorithm with depth limitation. Compared to level-wise, the advantage of leaf-wise is that for the same number of splits, leaf-wise

can reduce more error and achieve better accuracy. However, the disadvantage of leaf-wise is that it may grow very deep trees, leading to overfitting. Therefore, LightGBM adds a maximum depth limit on top of leaf-wise to prevent overfitting while ensuring high efficiency.

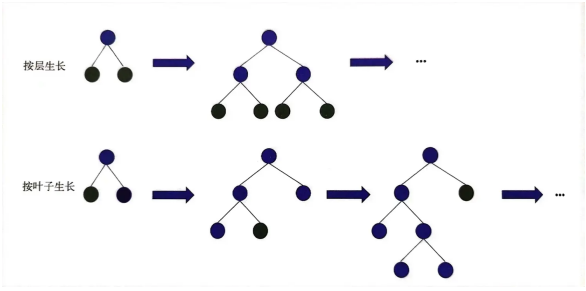

Figure 7: The left diagram shows the growth method of leaf nodes using the level-wise growth strategy. The right diagram illustrates the decision tree growth method using the leaf-wise growth strategy with depth limitation.The picture is from https://www.jianshu.com/p/25c661c84073

GOSS(Gradient-based One-Side Sampling)

In GBDT, data samples do not initially have weights. When calculating information gain, samples with different gradients have varying levels of importance. According to the definition of information gain, the larger the gradient, the greater the contribution to the calculation of information gain. Therefore, during training, the LightGBM model retains samples with large gradients and randomly samples some of the samples with small gradients (this maintains the original distribution of the data). This method provides a more accurate assessment than using the same sampling ratio for random sampling.

Exclusive Feature Bundling (EFB)

High-dimensional data often exhibit sparsity in their feature spaces, which provides an opportunity to reduce the number of features without loss of information. Specifically, in sparse feature spaces, many features are mutually exclusive (i.e., they do not simultaneously take non-zero values). Therefore, it is safe to bundle mutually exclusive features together. By employing a feature scanning algorithm, we can construct histograms using bundles of features that yield results comparable to those constructed using individual features.

Firstly, there are two questions. The first one is how to determine which features should be bundled together.

Finding the optimal bundling strategy is an NP-hard problem, meaning that we cannot find the optimal solution within a finite amount of time. Therefore, to find an approximate optimal solution, we transform the optimal bundling problem into a graph coloring problem, where each feature is treated as a node and non-exclusive nodes are connected by edges. We then use a greedy algorithm to solve it. However, we find that many features rarely take non-zero values simultaneously, even if they are not completely mutually exclusive. If the algorithm allows for minor conflicts among features, we can achieve fewer feature bundles and higher computational efficiency. This approach also has minimal impact on accuracy.

The second question is how to bundle the features.

To reduce the corresponding training complexity, we need a good method to merge these features into the same bundle. The key is to ensure that the values of the original features can be identified from the merged bundle. Since the histogram algorithm replaces continuous feature values with discrete bins, we distribute the mutually exclusive features into different bins to construct the bundle. This can be achieved by adding a bias constant to the feature values. For example, if we bundle two features, A and B, where the original values of feature A are in the interval [0, 10) and the original values of feature B are in the interval [0, 20), we can add a bias constant of 10 to the values of feature B, changing its range to [10, 30). The range of the bundled feature would then be [0, 30), allowing us to safely merge features A and B.

LightGBM Advantages

- Improved Training Speed: Utilizes the histogram algorithm to find the best split points during training, significantly enhancing training speed.

- Reduced Computation: Employs the gradient-based one-side sampling (GOSS) algorithm to reduce the computational load.

- Efficient Tree Construction: Uses the leaf-wise algorithm growth strategy to build decision trees, eliminating unnecessary computations.

- Lower Memory Consumption: Optimizations reduce the overall memory usage.

In conclusion, the LGBM algorithm can handle a variety of discrete features and is easy to train on CPUs. Therefore, we choose this algorithm for

ensemble learning.

Since the competition uses quadratic weighted kappa to evaluate the model, we use the gradient and hessian of the kappa function as the **objective function** when training LGBM.

$$
\begin{aligned}
\text{labels} &= y_{\text{true}} + a \\
\text{preds} &= (y_{\text{pred}} + a).\text{clip}(1, 6) \\
f &= \frac{1}{2}\sum_i (\text{preds}_i - \text{labels}_i)^2 \\
g &= \frac{1}{2}\sum_i (\text{preds}_i - a)^2 + 2b \\
\text{df} &= \text{preds} - \text{labels} \\
\text{dg} &= \text{preds} - a \\
\textbf{grad} &= \frac{\text{df}}{g} - \frac{f \cdot \text{dg}}{g^2} \cdot \text{len}(\text{labels}) \\
\textbf{hess} &= \text{ones}(\text{len}(\text{labels}))
\end{aligned}
$$

5 Experiments

We use the score on the public leaderboard as the evaluation metric. Moreover,We use confusion matrix to measure whether the model results converge near the correct thesis scores.

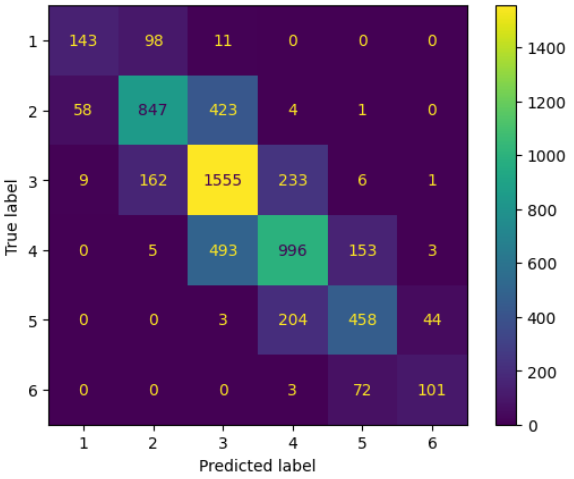

Figure 8: Confusion matrix of baseline

Although there are cases of misclassification, the misclassified scores are always concentrated near the correct scores, which actually verifies that articles with similar scores have similar characteristics.

In order to overcome overfitting, we have added k-fold cross-training on the basis of ensemble learning, using the average output of k ensemble learning models as the final output result.

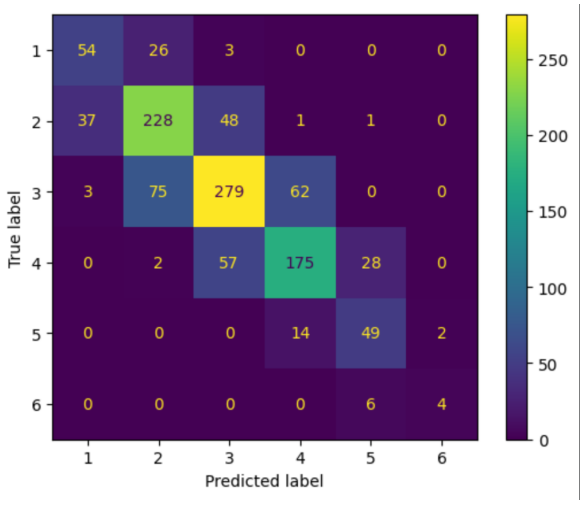

Figure 9: Confusion matrix of ensemble learning, one of the 15 folds

The confusion matrix for ensemble learning is shown as follows.

Experiments show that our methods achieve a better performance than the baseline model. And training with the additional data from Persuade 2.0 does not enhance the model's performance.

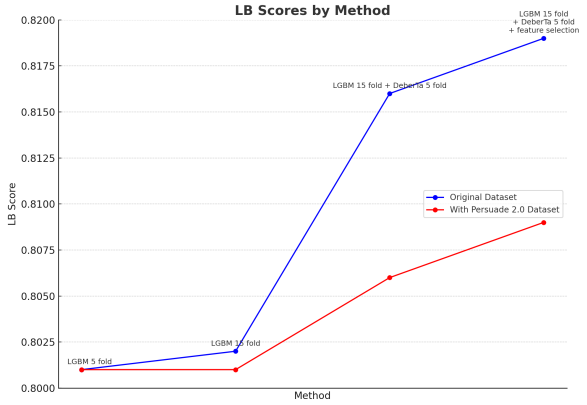

Figure 10: LB scores by method

After applying some data preprocessing techniques, our final model achieves a score of 0.824 on the public leaderboard, which is **top 1%** in the competition. The results can be found here, our team name is **Big Sword**.

6 Limitations

Ethic Problem

Machine learning-based models may encounter alignment issues in grading essays, such as a politically incorrect or morally questionable article being scored highly because its features resemble

those of high-scoring essays.Our research has not yet ventured into this area.

More powerful models

Clearly, it is promising to use models stronger than DeBERTa or BertGCN. In our research, we used the cls from an encoder-only model as the input to the softmax classification layer. In fact, we could also use a decoder-only model. In the future, we might utilize the API of GPT-4 to leverage its powerful zero-shot capabilities for score prediction.

More feature engineering

Features such as grammar feature, lexical feature, and syntactic feature are also worth exploring, as they can help us obtain a more complete representation of the entire article.

XGboost and LGBM may be better?

LightGBM and XGBoost each have unique advantages. LightGBM excels at handling large datasets and high-dimensional data, while XGBoost performs better in feature engineering and model tuning. By combining the two with weighted integration, maybe it can leverage their strengths, balancing the bias and variance of different models, thereby improving the model's generalization ability.

These shortcomings are the focus of our future efforts.

7 Acknowledgements

This work was made possible thanks to the meticulous instruction of Professor Zhou Baojian in the Natural Language Processing course at Fudan University's School of Data Science, as well as the computational resources (two 4090 GPU) provided by Professor Zhang Weizhong.

References

Veloso A Amorim E, Cançado M. 2018. Automated essay scoring in the presence of biased ratings. *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies*, page 229–237.

Cai C. 2019. Automatic essay scoring with recurrent neural network. *Proceedings of the 3rd international conference on high performance compilation, computing and communications*, page 1–7.

S. Dikli. 2006. Automated essay scoring. *Turk Online J Distance Educ*, pages 49–62.

Putri A et al Efendi T, Lubis FF. 2022. A bibliometrics-based systematic review on automated essay scoring in education. *Proceedings of the 2022 international conference on information technology systems and innovation*, page 275–82.

Sinharay S. Haberman SJ. 2010. The application of the cumulative logistic regression model to automated essay scoring. *Educ Behav Stat*, page 586–602.

Li P He Y, Chu X. 2023. Employing beautiful sentence evaluation to automatic chinese essay scoring. *Proceedings of the international conference on intelligent computing*, page 634–645.

Preet R et al Hendre M, Mukherjee P. 2023. Efficacy of deep neural embeddings-based semantic similarity in automatic essay evaluation. *Int J Cogn Inform Nat Intell IJCINI*, page 17(1):1–14.

D. Ifenthaler. 2022a. Automated essay scoring systems[m]//handbook of open, distance and digital education. *Singapore: Springer Nature Singapore*, pages 1–15.

D. Ifenthaler. 2022b. Automated essay scoring systems[m]//handbook of open, distance and digital education. *Singapore: Springer Nature Singapore*, pages 1–15.

Hui K et al Jin C, He B. 2018. Tdnn: a two-stage deep neural network for prompt-independent automated essay scoring. *Proceedings of the 56th annual meeting of the association for computational linguistics*, page 1088–97.

Gupta K. 2023. Data augmentation for automated essay scoring using transformer models. *Proceedings of the 2023 international conference on artificial intelligence and smart communication (AISC)*, page 3–7.

Ng V Ke Z. 2019. Automated essay scoring: a survey of the state of the art. *Proceedings of the IJCAI*, page 6300–6308.

Singh S et al Kumar Y, Parekh S. 2023. Automatic essay scoring systems are both overstable and oversensitive: explaining why and proposing defenses. *Dialogue Discourse*, page 14(1):1–33.

Kim M Lee J, Jang J. 2024. Hierarchical graph convolutional network approach for detecting low-quality. *Proceedings of the 2024 Joint International Conference on Computational Linguistics*, page 8108–8121.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. BertGCN: Transductive text classification by combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1456–1462, Online. Association for Computational Linguistics.

Guo J et al Ma C, Zhang Y. 2024. Fusionheightnet: A multi-level cross-fusion method from multi-source remote sensing images for urban building height. *Estimation*, page 16(6):958.

M. Nassef M.A. Hussein, H. Hassan. 2019. Automated language essay scoring systems: a literature review. *PeerJ Comput Sci*, page 208.

Black AW Mayfield E. 2020. Should you fine-tune bert for automated essay scoring? *Proceedings of the fifteenth workshop on innovative use of NLP for building educational applications*, page 151–62.

Holmes L Morris W, Crossley S. 2023. Using transformer language models to validate peer-assigned essay scores in massive open online courses (moocs). *13th international learning analytics and knowledge conference*, page 315–323.

Sasikala S et al Revathy JS, Maheswari NU. 2024. Automatic diagnosis of mental illness using optimized dynamically stabilized recurrent neural network. *Biomed Signal Process Control*, page 95:106321.

S.K. Mohamed S.M. Darwish. 2020. Proceedings of the international conference on advanced machine learning technologies and applications (amlta2019) 4. *Springer International Publishing*, pages 566–575.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. *Conference on Empirical Methods in Natural Language Processing*, page 1882–1891.

Cucchiarelli A Valenti S, Neri F. 2003. An overview of current research on automated essay grading. *Journal of Information Technology Education: Research*, page 319–330.

Uto M Yamaura M, Fukuda I. 2023. Neural automated essay scoring considering logical structure[c]. *International Conference on Artificial Intelligence*, page 267–278.

Li Y et al Yang K, Rakovi´c M. 2024. Unveiling the tapestry of automated essay scoring: A comprehensive investigation of accuracy airness and generalizability. *Proceedings of the AAAI Conference on Artificial Intelligence*, page 38(20):22466–74.

Jiao H Yao L. 2023. Comparing performance of feature extraction methods and machine learning models in essay scoring. *Chin Engl J Educ Meas Eval*, page 4(3):1.

Zhang Y Zhang Y, Cheng C. 2021. Multimodal emotion recognition using a hierarchical fusion convolutional neural network. *IEEE Access*, page 43–51.