NLP Course Final Project

Arseny Kazantcev

May 2024

Abstract

This document is dedicated to the final project of the Natural Language Processing course. In this project, I solved the problem of automatically evaluating essays. A link to my project code: `https://github.com/kazars24/nlp-course-final-project-itmo-2024`.

1 Introduction

The project aims to develop a solution for the "Learning Agency Lab - Automated Essay Scoring 2.0" competition from the Kaggle platform. Link to the competition: `https://www.kaggle.com/competitions/learning-agency-lab-automated-essay-scoring-2`.

The goal of this competition is to improve upon essay scoring algorithms to improve student learning outcomes[TLAL, 2024]. Efforts are needed to reduce the high expense and time required to hand grade these essays. Reliable automated techniques could allow essays to be introduced in testing, a key indicator of student learning that is currently commonly avoided due to the challenges in grading.

Essay writing is an important method to evaluate student learning and performance. It is also time-consuming for educators to grade by hand. Automated Writing Evaluation (AWE) systems can score essays to supplement an educator's other efforts. AWEs also allow students to receive regular and timely feedback on their writing. However, due to their costs, many advancements in the field are not widely available to students and educators. Open-source solutions to assess student writing are needed to reach every community with these important educational tools.

Previous efforts to develop open-source AWEs have been limited by small datasets that were not nationally diverse or focused on common essay formats. The first Automated Essay Scoring competition scored student-written short-answer responses, however, this is a writing task not often used in the classroom. To improve upon earlier efforts, a more expansive dataset that includes high-quality, realistic classroom writing samples was required. Further, to broaden the impact, the dataset should include samples across economic and location populations to mitigate the potential of algorithmic bias.

Competition host Vanderbilt University is a private research university in Nashville, Tennessee. For this competition, Vanderbilt has partnered with The Learning Agency Lab, an Arizona-based independent nonprofit focused on developing the science of learning-based tools and programs for the social good.

1.1 Team

Arseny Kazantcev prepared this document and the solution.

2 Related Work

In this section, I am going to describe in details the existing approaches to the problem.

The dataset proposed by the authors of the competition is not mentioned in any article. However, there are works in which different approaches were used on the dataset ASAP (Automated Student Assessment Prize). There are eight essay sets. Each of the sets of essays was generated from a single prompt. Selected essays range from an average length of 150 to 550 words per response. Some essays are dependent upon source information and others are not. All responses were written by students ranging in grade levels from Grade 7 to Grade 10. All essays were hand graded and were double-scored. Each of the eight data sets has its own unique characteristics. The variability is intended to test the limits of your scoring engine's capabilities.

Obviously, it would be incorrect to compare the results of the approaches trained on this dataset and the solution I developed, which was trained on the competition data. However, it is worth mentioning the approaches proposed by various authors for the task of automatic essay assessment.

2.1 Joint Learning of Multi-Scale Essay Representation

The work by Wang et al.[Yongjie Wang, 2022] suggest using multi-scale essay representation for BERT.

The model architecture is depicted in Figure 1.

The authors apply one BERT model to obtain the document-scale and token-scale essay representation. The concatenation of them is input into a dense regression layer, which predicts the score corresponding to the document-scale and token-scale. For each segment-scale k with number of segments m, they apply another BERT model to get m CLS outputs, and apply a $LSTM$ model followed by an attention layer to get the segment-scale representation. They input the segment-scale representation into another dense regression layer to get the score corresponding to segment-scale k. The final score is obtained by adding the scores of all S segment-scales and the score of the document-scale and token-scale, which is illustrated as below:

$$
\begin{aligned}
y &= \sum_k y_k + y_{doc,tok} \\
y_k &= \widehat{W}_{seg} \cdot o_k + b_{seg} \\
y_{doc,tok} &= \widehat{W}_{doc,tok} \cdot H_{doc,tok} + b_{doc,tok} \\
H_{doc,tok} &= w_{doc} \oplus W
\end{aligned}
$$

y_k is the predicted score corresponding to segment-scale k. $y_{doc,tok}$ is the predicted score corresponding to the document-scale and token-scale. $\hat{W}_{seg}$ and b_{seg} are weight matrix and bias for segment-scale respectively. $W_{doc,tok}$ and $b_{doc,tok}$ are weight matrix and bias for document and token-scales, ok is the segment-scale essay representation with the scale k. w_{doc} is the document-scale essay representation. W is the token-scale essay representation. $H_{doc,tok}$ is the concatenation of document-scale and token-scale essay representations.

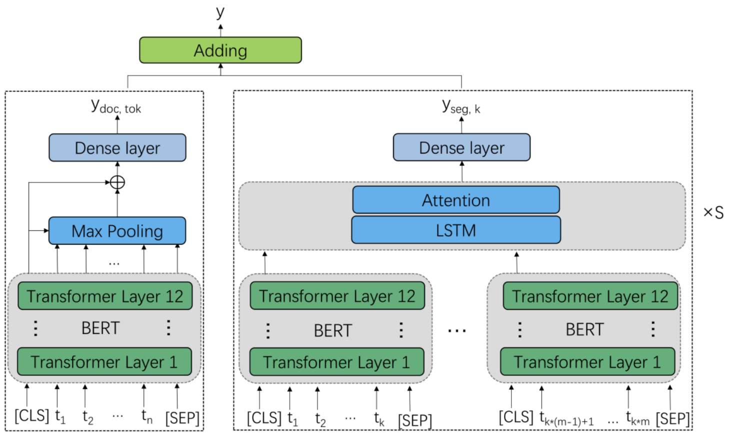

Figure 1: The proposed automated essay scoring architecture based on multi-scale essay representation. The left part illustrates the document-scale and token-scale essay representation and scoring module, and the right part illustrates S segment-scale essay representations and scoring modules.

2.2 Two-Stage Learning

The authors of the paper [Jiawei Liu, 2019] propose an approach to Automated Essay Scoring by developing a Two-Stage Learning Framework (TSLF). This approach integrates the strengths of both feature-engineered models and end-to-end models to enhance the robustness and effectiveness of AES systems.

In the first stage, the framework calculates three types of scores using deep neural networks:

1. Semantic Score: This score evaluates the essay based on its semantic content, leveraging deep semantic information. This is achieved using LSTM networks which process the essay to capture its overall meaning.
2. Coherence Score: Designed to detect adversarial samples with permuted paragraphs that are well-written but semantically incoherent.
3. Prompt-relevant Score: Aimed at identifying essays that are irrelevant to the given prompt.

These scores are designed to handle adversarial samples effectively, ensuring the AES system is robust against well-crafted but misleading essays.

In the second stage, the scores from the first stage are concatenated with handcrafted features, such as the number of words and grammar errors. This combined feature set is then fed into a boosting tree model (specifically XGBoost) for further training. This hybrid approach allows the model to leverage both the detailed, explainable handcrafted features and the comprehensive, deep semantic features captured by the neural networks.

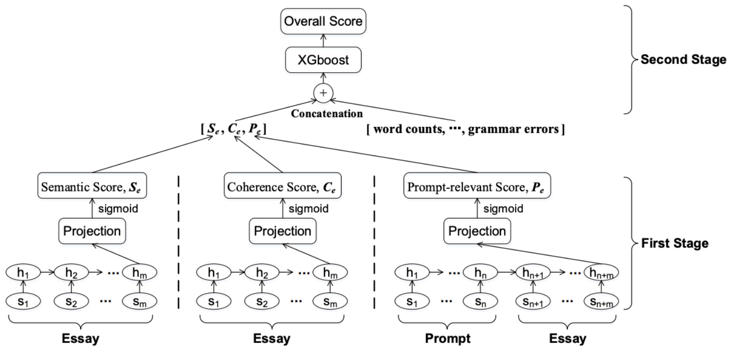

Figure 2: Two-Stage Learning Framework for AES. In the first stage, based on deep neural networks, we calculate semantic score, coherence score and prompt-relevant score named as S_e, C_e and P_e respectively. C_e and P_e are proposed to detect adversarial samples. In the second stage, we concatenate these three scores with some handcrafted features and feed the result to the boosting tree model for further training.

2.3 Considering-Content-XLNet

The authors of the paper [Sungho Jeon, 2021] note that previous machine learning-based AES systems often rely on essay length as a proxy for essay quality, a practice criticized for not accurately reflecting writing proficiency. This problem persists even in recent neural essay scoring systems, which, despite not explicitly including essay length as a feature, still show a strong correlation between essay length and assigned scores.

The authors propose a neural model that considers essay content rather than length. This model assesses the similarity of word distributions between the input essay and essays grouped into three score levels: low, mid, and high. The model calculates the Kullback-Leibler (KL) divergence between these distributions, allowing it to focus on the content quality rather than length. The KL divergence for an input essay x is defined as:

$$KL(p_{\text{lvl}}, q) = \sum_{x} p_{\text{lvl}}(x) \log \left(\frac{p_{\text{lvl}}(x)}{q(x)} \right)$$

where p_{lvl} is the word distribution in the essays grouped at level lvl and q is the word distribution in the input essay.

The authors conclude that neural essay scoring systems should account for the characteristics of the dataset and focus on text quality rather than length. Their proposed content-based model using KL divergence effectively reduces the influence of essay length and improves the system's ability to replicate human scoring more accurately.

2.4 String kernels and word embeddings

The paper [Mădălina Cozma and Ionescu, 2018] presents an approach combined string kernels and word embeddings. This method aims to improve the accuracy and reliability of automated grading systems used in educational settings. Their method is designed to leverage both low-level character n-gram features and high-level semantic features to achieve state-of-the-art performance.

String kernels are used to capture the similarity among strings based on counting common character n-grams. The process involves the following steps:

1. They extract character n-grams (sequences of n characters) from the essays. These n-grams serve as low-level features that capture syntactic patterns.

2. The specific string kernel used is the histogram intersection string kernel (HISK), which calculates the similarity between two strings x and y as:

   $$k_{\cap}(x, y) = \sum_{v \in \Sigma^n} \min(\text{num}_v(x), \text{num}_v(y))$$

   where Σ is the alphabet, v is an n-gram, and $\text{num}_v(x)$ is the number of occurrences of v in x.

3. They construct a kernel matrix where each entry represents the similarity between a pair of essays based on the HISK.

Word embeddings capture high-level semantic information by representing words as vectors in a continuous vector space. The steps involved are:

1. They use pre-trained word embeddings (e.g., word2vec) to obtain vector representations of words in the essays.

2. They use Bag-of-Super-Word-Embeddings (BOSWE) which is a new representation that combines word embeddings into higher-level features:

   - Clustering: Word vectors are clustered using k-means clustering to form semantic clusters.
   - Super Word Embeddings: Each cluster centroid is considered a super word embedding representing a group of semantically related words.
   - Document Representation: Essays are represented as vectors indicating the occurrence counts of super word embeddings in the essay.

The authors combine the two feature sets (string kernels and BOSWE) to create a comprehensive representation of the essays:

1. In the dual form, the kernel matrices from HISK and BOSWE are summed to form a single combined kernel matrix. This approach implicitly concatenates the feature vectors in the primal space.

$$K_{+} = K_{\text{HISK}} + K_{\text{BOSWE}}$$

2. They use v-Support Vector Regression (v-SVR) to train the model on the combined kernel matrix. v-SVR is a type of Support Vector Machine used for regression tasks.

The combination of string kernels and word embeddings effectively captures both syntactic and semantic information, leading to significant improvements in automated essay scoring. The approach outperforms deep learning models and demonstrates the value of integrating different levels of textual features.

3 Model Description

My approach integrates various preprocessing steps, feature engineering techniques, and combines predictions from 2 models.

The high-level architecture diagram is shown in Fig. 3.

3.1 Paragraph, Sentence, and Word Preprocessing

The text data undergoes preprocessing at three levels:

- Paragraph Preprocessing: This step involves analyzing the essay at the paragraph level. Features such as the number of paragraphs and the length of each paragraph are extracted.

- Sentence Preprocessing: The essay is then analyzed at the sentence level. This includes counting the number of sentences, and extracting various sentence-level features like average sentence length.

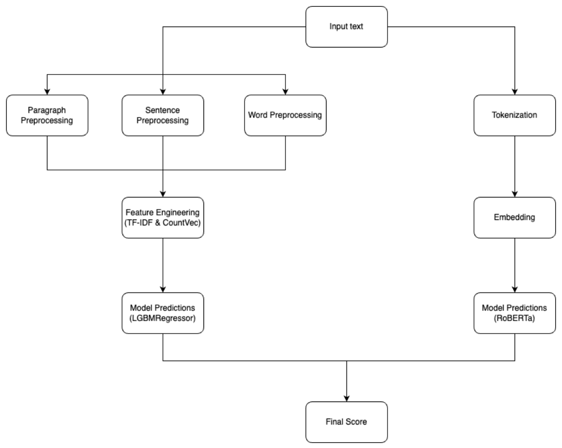

Figure 3: The provided diagram illustrates the architecture of an automated essay scoring (AES) system. The system combines various preprocessing steps, feature engineering techniques, and model predictions to generate a final score for an input text.

- Finally, the essay is analyzed at the word level. This involves counting the number of words, and extracting word-level features such as the average word length and the presence of specific keywords.

Each level involves the following steps:

1. Convert words to lowercase;
2. Remove HTML;
3. Delete strings starting with @;
4. Delete Numbers;
5. Delete URL;
6. Replace consecutive empty spaces with a single space character;
7. Replace consecutive commas and periods with one comma and period character;
8. Remove empty characters at the beginning and the end.

3.2 TF-IDF and Count Vectorizer

The approach applies both TF-IDF and Count Vectorizer to the text data to create additional features.

Firstly, the text data is transformed into TF-IDF vectors, which are then converted into a dense matrix and merged with other features. TF-IDF formula:

$$\mathrm{tf}(t,d) = \frac{\text{count of } t \text{ in } d}{\text{total terms in } d}$$

$$\mathrm{idf}(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

where N is the total number of documents in the corpus and $|\{d \in D : t \in d\}|$ is the number of documents containing the term t.

Secondly, Count Vectorizer transforms the text into numerical features based on the frequency of each word in the text.

3.3 Tokenization and Embedding

The text is tokenized into smaller units (tokens) that are understood by the RoBERTa model. Tokenization breaks down the text into subwords. The tokenized text is converted into embeddings, which are numerical representations of the text that capture its semantic meaning. These embeddings are used as input to the RoBERTa model.

3.4 Gradient Boosting Model (LGBMRegressor)

Gradient Boosting is an ensemble machine learning technique that builds models sequentially, where each new model attempts to correct the errors of the previous ones. LightGBM (Light Gradient Boosting Machine) is a highly efficient implementation of gradient boosting that is designed to be distributed and efficient with large datasets.

In my automated essay scoring system, LGBMRegressor is used to predict essay scores. The model is trained on a labeled dataset of essays, where it learns to predict the score based on the textual features extracted during preprocessing.

3.5 RoBERTa

RoBERTa (Robustly Optimized BERT Pre-training Approach) [Yinhan Liu, 2019] is an advanced variant of the BERT (Bidirectional Encoder Representations from Transformers) model, designed to improve upon the limitations of BERT. RoBERTa enhances the performance by using more data and computational resources and by modifying the training procedure.

In my automated essay scoring system, RoBERTa is used to predict essay scores based on the semantic understanding of the text. The model is fine-tuned on a labeled dataset of essays to capture the nuances and intricacies of essay writing.

3.6 Combining Model Predictions

The final score for the essay is calculated by averaging the predictions from the LGBMRegressor and the RoBERTa model:

$$\text{final\_score} = \frac{\text{score}_{\text{LGBM}} + \text{score}_{\text{RoBERTa}}}{2}$$

4 Dataset

The Learning Agency Lab Automated Essay Scoring dataset, sourced from Kaggle, serves as a benchmark for developing automated essay scoring (AES) systems. With a focus on educational assessment, this dataset enables researchers and practitioners to explore the application of machine learning techniques in evaluating essays. The competition dataset comprises about 24000 student-written argumentative essays. Each essay was scored on a scale of 1 to 6.

To obtain the dataset for the Learning Agency Lab Automated Essay Scoring competition on Kaggle, follow these steps:

1. Visit the competition page.
2. If you don't already have a Kaggle account, you'll need to sign up for one.
3. Once logged in, navigate to the "Data" tab on the competition page.
4. You'll find the dataset files available for download. Click on the dataset you're interested in to access it.
5. Download the dataset files to your local machine.

The dataset comprises three primary files:

- train.csv - 17307 essays and scores to be used as training data.
  - essay_id - The unique ID of the essay
  - full_text - The full essay response
  - score - Holistic score of the essay on a 1-6 scale
- test.csv - 8000 essays to be used as test data. Contains the same fields as train.csv, aside from exclusion of score.
- sample_submission.csv - A submission file in the correct format.
  - essay_id - The unique ID of the essay
  - score - The predicted holistic score of the essay on a 1-6 scale

5 Experiments

5.1 Metrics

The metric proposed for evaluating the models in the competition "Learning Agency Lab - Automated Essay Scoring 2" is the Quadratic Weighted Kappa (QWK).

Quadratic Weighted Kappa is a metric that measures the agreement between two raters. In the context of this competition, it is used to assess the similarity between the predicted essay scores and the actual scores given by human graders. The QWK metric ranges from -1 to 1, where:

- 1 indicates perfect agreement;
- 0 indicates no agreement (equivalent to random chance);
- Negative values indicate agreement worse than random chance.

The QWK is computed through the following steps:

1. Confusion Matrix (O): Construct a confusion matrix O where O_{ij} represents the number of essays that received a score of i from the human grader and a score of j from the model.
2. Expected Matrix (E): Calculate the expected matrix E assuming that the scores from the human grader and the model are independent. This is computed as:
$$E_{ij} = \frac{N_{i\cdot} \times N_{\cdot j}}{N}$$
where $N_{i\cdot}$ is the sum of the i-th row, $N_{\cdot j}$ is the sum of the j-th column, and N is the total number of essays.
3. Weight Matrix (W): Define a weight matrix W where the weight for each cell (i, j) is calculated as:
$$W_{ij} = \frac{(i-j)^2}{(N-1)^2}$$
This weights the disagreements based on the squared difference between the actual and predicted scores.
4. QWK Calculation: The QWK is then computed using the formula:
$$\kappa = 1 - \frac{\sum_{i,j} W_{ij} O_{ij}}{\sum_{i,j} W_{ij} E_{ij}}$$
5.

QWK is particularly suitable for essay scoring for the following reasons:

1. Sensitivity to Agreement: It accounts for the degree of disagreement between scores, giving higher penalties for larger differences.
2. Alignment with Human Judgment: Since essay scoring is often subjective, QWK aligns well with the way human graders perceive agreement.
3. Normalization: The metric normalizes the agreement by considering the expected agreement, making it robust against random guessing.

The use of QWK in the "Learning Agency Lab - Automated Essay Scoring 2" competition ensures that models are evaluated on their ability to produce scores that closely match human judgment, thereby fostering the development of more accurate and reliable automated essay scoring systems.

5.2 Experiment Setup

The experiment began with data preprocessing, which was described earlier. Thanks to feature engineering, the data preprocessed for gradient boosting contained 31753 features. Preprocessed data was divided into training and validation sets by sklearn's train_test_split with test_size=0.2, random_state=52 and stratification by score.

LGBMRegressor was used as a gradient boosting model with:

- learning_rate = 0.05,
- max_depth = 5,
- num_leaves = 10,
- colsample_bytree=0.3,
- reg_alpha = 0.7,
- reg_lambda = 0.1,
- n_estimators=700,
- random_state=42,
- extra_trees=True,
- class_weight='balanced',
- the other hyperparameters had default values.

Pre-trained RoBERTa model was deepset/roberta-base-squad2-distilled. This model has been fine-tuned with:

- per_device_train_batch_size=16,
- per_device_eval_batch_size=16,

- num_train_epochs=3,
- warmup_steps=500,
- the other hyperparameters had default values.

5.3 Baselines

Since the dataset of the competition was not mentioned or used in other articles, I chose logistic regression over TF-IDF embedding as the baseline.

The data was also divided into training and validation sets by sklearn's train_test_split with test_size=0.2, random_state=52 and stratification by score. After that, the English stop words were deleted. This text data was transformed into numerical features using TfidfVectorizer. This converts the text into a matrix of TF-IDF features, which represent the importance of terms in the documents. A logistic regression model (LogisticRegression from scikit-learn) was trained on the TF-IDF features of the training data. This approach achieved a QWK = 0.437 in the test sample.

The implementation of gradient boosting from CatBoost without any additional text preprocessing and feature engineering achieved a QWK = 0.718 in the test sample. To do this, we used built-in methods for converting text features into numeric ones, for example BoW (Bag of words).

6 Results

To begin with, I will give the results of the various approaches that I have tried.

Also in this section, you could provide some results for your model inference. The samples could be found in Tab. 1.

Approach	QWK
LogReg + TF-IDF	0.437
CatBoostClassifier	0.718
BERT	0.751
RoBERTa	0.764
GB + RoBERTa	0.790

Table 1: The combination of Gradient Boosting and fine-tuned RoBERTa achieved the highest metric value among all my solutions.

My approach turned out to be better than baseline and some other solutions, but there are also approaches that showed the value of the target metric higher than mine. The approaches and their metrics can be viewed in Tab. 2.

Speaking of competitive leaderboard, I ranked 911 out of 1,573 on the public part of the test data. Due to capacity limitations, it was not possible to train more complex and large models and achieve a better result.

Approach	QWK
Modeling with FB scoring	0.822
GB + RoBERTa	0.790
KerasNLP	0.752
ASE - FIGHTING	0.742
DeBERTa	0.718

Table 2: Different approaches suggested by other participants of the competition. All titles are clickable and lead to pages with the solution code

7 Conclusion

In conclusion, this project successfully tackled the challenge of automated essay scoring by integrating advanced NLP models, comprehensive preprocessing, and robust evaluation techniques. The developed solution shows promising results, making a significant contribution to the field of automated educational assessment.

References

[Jiawei Liu, 2019] Jiawei Liu, Yang Xu, Y. Z. (2019). Automated essay scoring based on two-stage learning.

[Mădălina Cozma and Ionescu, 2018] Mădălina Cozma, A. M. B. and Ionescu, R. T. (2018). Automated essay scoring with string kernels and word embeddings.

[Sungho Jeon, 2021] Sungho Jeon, M. S. (2021). Countering the influence of essay length in neural essay scoring. *EMNLP*.

[TLAL, 2024] TLAL (2024). Automated essay scoring 2.0. *Kaggle*.

[Yinhan Liu, 2019] Yinhan Liu, Myle Ott, N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. (2019). Roberta: A robustly optimized bert pretraining approach.

[Yongjie Wang, 2022] Yongjie Wang, Chuan Wang, R. L. H. L. (2022). On the use of bert for automated essay scoring: Joint learning of multi-scale essay representation. *NAACL*.