

Automated Essay Scoring

1. Overview

Problem Definition

The goal of this project is to train a model to score student 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.

Competition webpage

<https://www.kaggle.com/competitions/learning-agency-lab-automated-essay-scoring-2>

Metric

The metric is 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 $N \times N$ confusion matrix O is constructed, such that $O_{i,j}$ corresponds to the number of `essay_ids` i (actual) that received a predicted value j . An $N \times N$ 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 $N \times N$ 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:

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

A comprehensive explanation with examples is available at:

<https://www.kaggle.com/code/aroraaman/quadratic-kappa-metric-explained-in-5-simple-steps>

Data

train.csv

The file `train.csv` contains about 17,000 full-text, student-written argumentative essays on various topics. Each essay is scored on a scale of 1 to 6. [The scoring system](#) primarily evaluates the underlying meaning of an essay and the student's ability to clearly demonstrate critical thinking and reasoning.

Columns:

- o `essay_id` - the unique ID of the essay;
- o `full_text` - the full essay text;
- o `score` - score of the essay on a 1-6 scale.

Strategy for solving the problem

1. Develop a baseline solution using a tree-based model that utilizes quantitative features from essays, such as word count, paragraph count, and number of grammatical errors.
2. Fine-tune a pre-trained large language model using the transformers library.

2. Training Data

Target







Source code

`01_LGB_baseline.ipynb` – a jupyter notebook with basic text statistics on essays and a baseline LGB solution on text statistical features;

`DataExplorer.py` – a class `DataExplorer` for exploratory data analysis on tabular data.

Overview

Training dataset contains 17,307 rows with full texts of essays and their corresponding scores on a scale from 1 to 6. Distribution of scores is as follows:

		count
score		
1		1252
2		4723
3		6280
4		3926
5		970
6		156

Linguistic features

Source code

`02_grammatical_error_count.ipynb` – a jupyter notebook for computing grammatical features;
`03_language_features.ipynb` – a jupyter notebook with EDA on grammatical features;
`LanguageFeaturesExtractor.py` – a class `LanguageFeaturesExtractor` for assessing grammar and punctuation of texts.

Overview

The `LanguageFeaturesExtractor` class is designed to extract and analyze language-related errors from text data using the `LanguageTool` library. The class focuses on identifying various types of linguistic errors, categorizing them, and counting their occurrences within the text.

Examples of linguistic features are a count of errors for a given text:

- number of typos,
- number of punctuation errors,
- number of confused words etc.

Baseline LGB out-of-fold metric for 5 folds

Two baseline LGB models were estimated on linguistic features:

1. Count of each error type,
2. Frequency of each error type (count divided by total word count).

The out-of-fold scores of both models fall within the same range.

Fold	Error count		Error frequency	
	Quadratic weighted kappa score	Best iterations	Quadratic weighted kappa score	Best iterations
0	0.711	43	0.702	42
1	0.699	46	0.700	41
2	0.692	44	0.691	38
3	0.704	39	0.709	41
4	0.708	40	0.706	37

Statistical features

Source code

`04_text_features.ipynb` – a jupyter notebook with EDA on the text quantitative features and baseline LGB solution using grammatical and text quantitative features;
`TextFeaturesExtractor.py` – a class `TextFeaturesExtractor` for computing quantitative features of texts.

Overview

The `TextFeaturesExtractor` class computes a variety of statistical features for paragraphs, sentences, and words in a text.

Features	Count-based Statistics	Descriptive Statistics
Paragraph	<ul style="list-style-type: none">Number of paragraphs with lengths greater than or equal to specific values (0, 50, 75, 100, 125, 150, 175, 200, 250, 300, 350, 400, 500, 600, 700).Number of paragraphs with lengths less than or equal to specific values (25, 49).	<i>Maximum, mean, minimum, sum, first, last, kurtosis, 1st quartile (Q1), and 3rd quartile (Q3) for:</i> <ul style="list-style-type: none">Paragraph length,Paragraph sentence count,Paragraph word count,Paragraph error number.
Sentence	<ul style="list-style-type: none">Number of sentences with lengths greater than or equal to specific values (0, 15, 50, 100, 150, 200, 250, 300).Number of sentences with lengths less than or equal to specific values (15, 50).	<i>Maximum, mean, minimum, sum, first, last, kurtosis, 1st quartile (Q1), and 3rd quartile (Q3) for:</i> <ul style="list-style-type: none">Sentence length,Sentence word count.
Word	<ul style="list-style-type: none">Number of words with lengths greater than or equal to specific values (1 to 15).	<i>Maximum, mean, standard deviation, 1st quartile (Q1), 2nd quartile (Q2), and 3rd quartile (Q3) for</i> <ul style="list-style-type: none">word lengths.

Baseline LGB out-of-fold metric for 5 folds

Fold	Quadratic weighted kappa score	Best iterations
0	0.707	42
1	0.698	41
2	0.691	33
3	0.707	42
4	0.712	45

The scores are within the same range as the scores on linguistic features.

Baseline LGB out-of-fold metric for 5 folds on linguistic and statistical features

The results of a baseline LGB model using all linguistic and statistical features are shown in the table below:

<i>Fold</i>	<i>Quadratic weighted kappa score</i>	<i>Best iterations</i>
0	0.715	46
1	0.711	46
2	0.699	41
3	0.720	44
4	0.720	40

Concatenating text and linguistic features only marginally improves the out-of-fold baseline scores.

3. LLM solution

Solution with HuggingFace transformers tools

Source code

`05_transformers.ipynb` – a jupyter notebook for training a DeBERTa-v3-xsmall model on one fold of the data split;

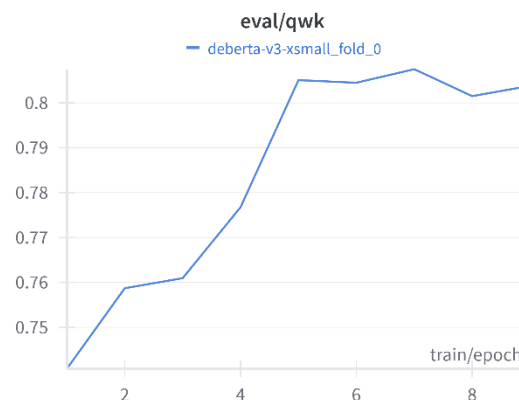
`transformers_trainer.py` – a wrapper class `TransformersClassifier` that provides functionality for training and predicting with a HuggingFace transformers classifier model.

Overview

The `TransformersClassifier` class streamlines the process of training and evaluating transformer-based models for this project. It handles tokenization, model configuration, training, evaluation, and prediction, with support for logging and tracking metrics through wandb.

Examples of training and prediction are in the notebook `05_transformers.ipynb`.

The best quadratic Cohen kappa score for fold 0 (with 5-fold split of the dataset) is **0.807**, which significantly exceeds the results of the tree-based solutions. The evolution of the validation score during training is shown in the chart below:



Custom solution

Source code

`ES_dataset.py` – a class `EssayDataset` that prepares the data in the right format for the PyTorch model to work with;

`ES_model.py` – a class `EssayClassifierModel` that provides a custom classifier head for a pre-trained LLM;

`experiment.py` – provides a high-level functionality for training a PyTorch model;

`training_loop.py` – provides a training loop for a PyTorch model;

`06_train_custom.ipynb` – a jupyter notebook for training a custom PyTorch model for this project.

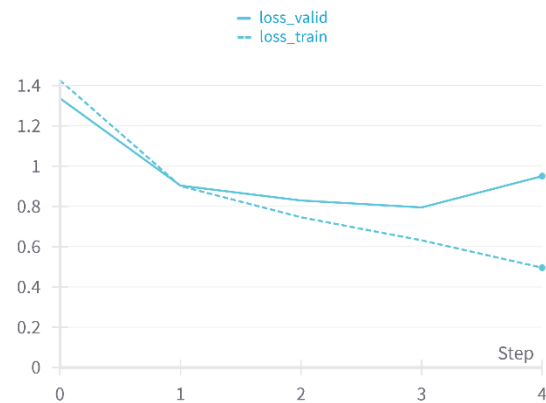
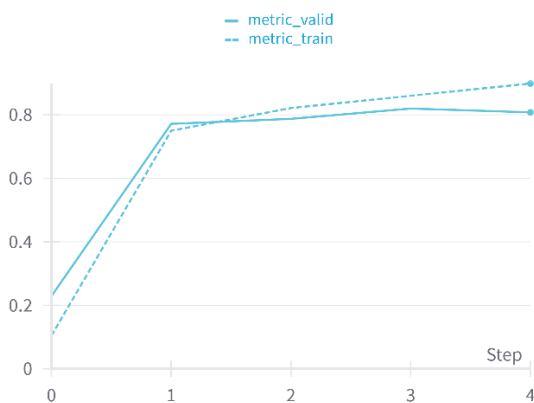
Overview

The `EssayClassifierModel` provides a custom classifier head for a pre-trained LLM model from transformers library.

The data is split into 5 folds, validation is performed on fold 0. The parameter values for the best training example are:

- `deberta-v3-small` model checkpoint;
- AdamW optimizer;
- `CrossEntropyLoss`;
- For the first epoch only the classifier head is trained, other parameters are frozen;
- Exponential learning rate scheduler with $\gamma=0.8$, starting learning rate is 10^{-5} .

The evolution of loss and metrics can be seen in the charts below. **The best validation metric is 0.8198** at epoch 3. After that, the model starts to overfit.



4. Areas of improvement

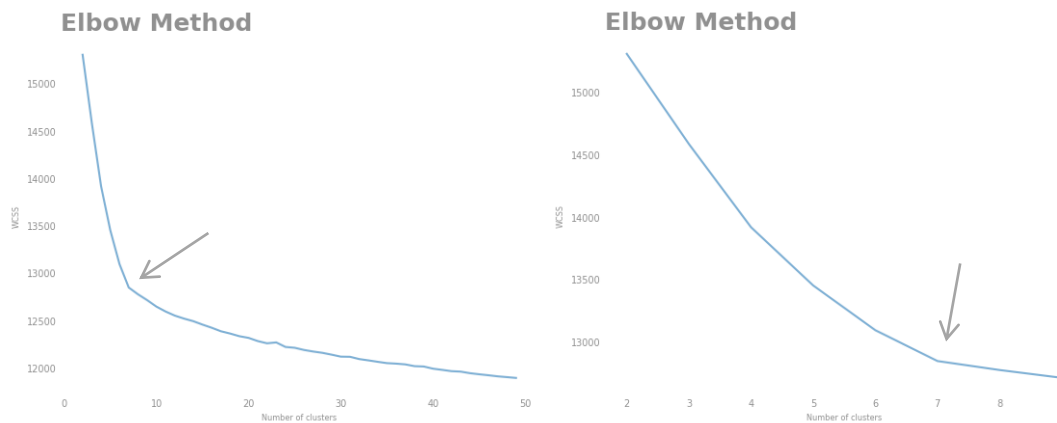
Clusters

Source code

07_clusterization.ipynb – a jupyter notebook for determining the number of clusters;
08_7_clusters.ipynb – a jupyter notebook for exploring clusters;
09_score_by_clusters.ipynb – a jupyter notebook for exploring OOF predictions by cluster;
10_cluster3_fold0.ipynb ipynb – a jupyter notebook for exploring OOF predictions for cluster 3.

Determining the number of clusters

The dataset contains essays on various topics. With TFIDF encoding of the essays and KMeans clustering, the elbow method indicates that **there are seven distinct essay topics** in the dataset. This is illustrated in the charts below:



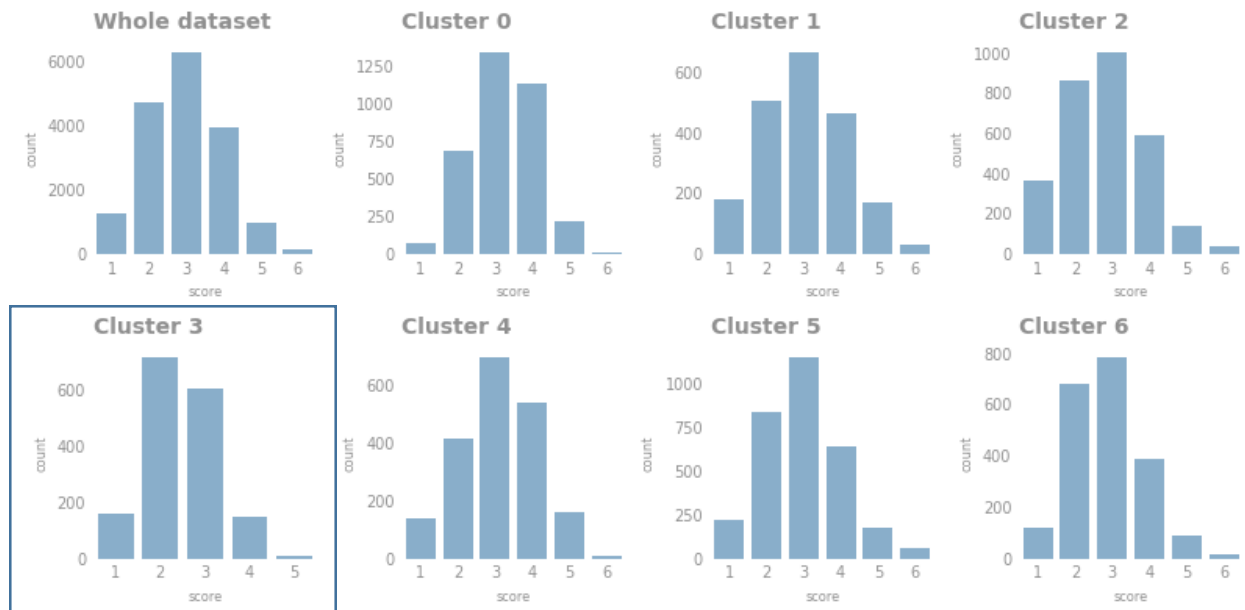
In the notebook 08_7_clusters.ipynb it is shown that assuming the number of clusters as 8 results in splitting one of the 7 original clusters, while the other clusters remain unchanged.

Distribution of target by cluster

The distribution of cluster sizes is as follows:

Cluster	Size
0	20.16%
1	11.70%
2	17.33%
3	9.52%
4	11.36%
5	17.86%
6	12.06%

The distribution of essay scores for each cluster:



Note: Cluster 3 is the least populated and has a very different distribution of the target score compared to the rest of the clusters.

Stability of clusters

As shown in the notebook `08_7_clusters.ipynb`, the result of clusterization is the same for the whole dataset and for out-of-fold clusterization for each of 5 folds. This is illustrated with confusion matrices below:

Figure 1 displays four heatmaps showing the distribution of cluster sizes for the whole dataset and out-of-fold clusters across four folds (Fold 0, Fold 1, Fold 2, Fold 4). The y-axis for each heatmap represents the cluster size, and the x-axis represents the frequency of clusters of that size.

Fold 0: The whole dataset cluster size distribution is shown on the left, and the out-of-fold cluster size distribution is on the right. The whole dataset distribution is highly skewed towards smaller cluster sizes (0-100), while the out-of-fold distribution is more spread out, peaking around 300-400.

Fold 1: The whole dataset cluster size distribution is shown on the left, and the out-of-fold cluster size distribution is on the right. The whole dataset distribution is highly skewed towards smaller cluster sizes (0-100), while the out-of-fold distribution is more spread out, peaking around 300-400.

Fold 2: The whole dataset cluster size distribution is shown on the left, and the out-of-fold cluster size distribution is on the right. The whole dataset distribution is highly skewed towards smaller cluster sizes (0-100), while the out-of-fold distribution is more spread out, peaking around 300-400.

Fold 4: The whole dataset cluster size distribution is shown on the left, and the out-of-fold cluster size distribution is on the right. The whole dataset distribution is highly skewed towards smaller cluster sizes (0-100), while the out-of-fold distribution is more spread out, peaking around 300-400.

Distribution of out-of-fold predictions by cluster for fold 0

Fold 0 (of 5-fold split) has a similar distribution of cluster sizes to the distribution on the whole dataset.

Quadratic Cohen kappa score for out-of-fold predictions of the fine-tuned Deberta-v3-small model (see Section 3, Custom model) is 0.8198. The metrics for those predictions for each cluster are shown in the table below:

Cluster	Size	Quadratic Cohen kappa score
0	20.28%	0.7857
1	12.28%	0.8359
2	18.33%	0.8299
3	8.69%	0.6991
4	11.27%	0.7795
5	17.04%	0.8472
6	12.22%	0.7831

Cluster 3 metric is significantly lower than the rest of the clusters.

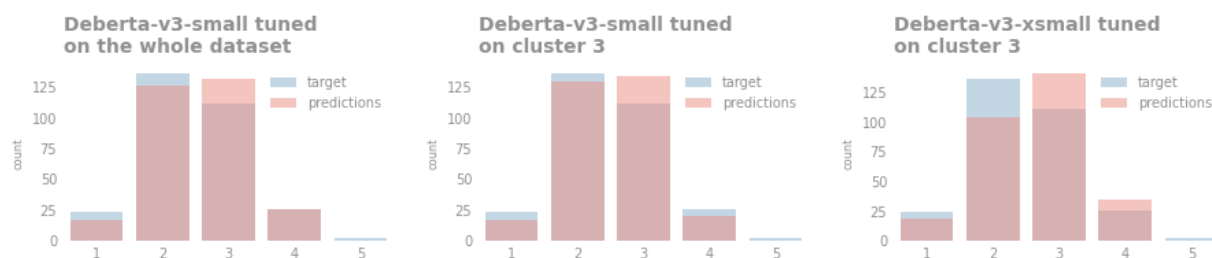
Cluster 3 training for fold 0

Re-training cluster 3 for fold 0 (out of 5 folds):

Model	Quadratic Cohen kappa score
Deberta-v3-small trained on the whole dataset	0.6991
Deberta-v3-small trained on cluster 3	0.6881
Deberta-v3-xsmall trained on cluster 3	0.6853

Note: applying the same training schedule to cluster 3 doesn't produce good results in terms of quadratic Cohen kappa score.

Also, training models only on cluster 3 doesn't improve the distributions of predictions:



Conclusion

While the baseline solution for cluster 3 hasn't shown an improvement in the metric, this approach has not been fully explored and holds potential for future enhancement.

Due to available hardware limitations, more computationally intensive experiments were not conducted.