Linking writing behaviour with writing quality

Neeraj Gupta
negupt@iu.edu

Dhiraj Pimparkar
dkpimpar@iu.edu

Ayush Manojkumar Lodha
amlodha@iu.edu

Abstract

Our project, derived from a Kaggle competition hosted by "The Learning Agency Lab," investigates the influence of typing behavior on the quality of written essays. Our approach involved extensive feature engineering, enhancing the original dataset from 11 to 146 features by including keystroke dynamics like 'up_event' counts and summarizing behavioral patterns. We employed Principal Component Analysis (PCA) to reduce the feature set to 30, retaining 95% of the variance for analysis. Our experimental setup included various machine learning models: Support Vector Classification (SVC) for categorizing essays, Light Gradient Boosting (LGB) Regressor for a regression approach, a Multi-layer Perceptron (MLP) as a deep learning model, and Transformers for advanced text analysis. The LGB Regressor emerged as the most effective, significantly outperforming our benchmark in Root Mean Square Error (RMSE), a key metric for this task. This study provides new insights into how typing behavior correlates with essay quality, suggesting potential applications in educational technology and writing assessment tools.

1 Introduction

1.1 Problem Statement

The primary question guiding our research was: How does typing behavior impact the quality of written essays? The challenge lay in capturing and analyzing the nuanced aspects of typing behavior, such as pause patterns, revision strategies, and overall keystroke dynamics. We aimed to understand these complexities by developing predictive models using a dataset of keystroke logs. The project's novelty stemmed from its focus on the writing process rather than just the final product, a common limitation in traditional writing assessments. Our goal was to identify patterns in typing behavior that correlate with higher-quality writing, thereby contributing to the development of more sophisticated writing instruction and assessment methodologies.

This comprehensive approach, aligning with the objectives of the ongoing Kaggle competition, provides a fresh perspective on assessing writing quality through data science methodologies.

1.2 Data overview

The dataset used in this project is based on keystroke logging information, specifically designed to analyze the impact of typing behavior on the quality of written essays. The dataset that we will be working with consists of two tables: sequential logs data and scores for each essay (train_logs and train_score). The first table, train_logs, has the data on the actions that were performed while the essay was being written. For example, when the user pressed a key, whether he deleted or inserted any character, etc. The second table, train_score, has scores awarded to that essay.

The initial dataset comprises a collection of 8,405,898 data instances, distributed across 2,471 essays(Refer to table 1). Each essay is uniquely identified through a series of event IDs, allowing for individualized tracking and analysis. It is essential to note that, to ensure privacy and data security, specific characters entered by the user within the dataset are anonymized and represented by the placeholder "q" rather than their actual values. This anonymization approach safeguards sensitive information, making it impossible to reconstruct the original essays, and preserves the confidentiality of each user.

A detailed description of available features is provided in the table

	train_log.csv	train_score.csv
No of Samples	8,405,898	2471
Total Number of Columns	11	2

Table 1: Data size overview

Event ID	Down Time	Up Time	Action Time	Event	Position	Word Count	Text Change	Activity
1	30185	30395	210	Leftclick	0	0	NoChange	Nonproduction
2	41006	41006	0	Shift	0	0	NoChange	Nonproduction
3	41264	41376	112	I	1	1	I	Input
4	41556	41646	90	Space	2	1		Input
5	41815	41893	78	b	3	2	b	Input
6	42018	42096	78	e	4	2	e	Input
7	42423	42501	78	l	5	2	l	Input
8	42670	42737	67	i	6	2	i	Input
9	42873	42951	78	e	7	2	e	Input
10	43041	43109	68	v	8	2	v	Input
11	43289	43378	89	Space	9	2		Input
12	44560	44605	45	Backspace	8	2		Remove/Cut
13	44661	44762	101	e	9	2	e	Input
14	44954	45032	78	Space	10	2		Input
15	45325	45381	56	t	11	3	t	Input
16	45460	45538	78	h	12	3	h	Input
17	45640	45730	90	a	13	3	a	Input
18	45741	45808	67	t	14	3	t	Input
19	45933	46011	78	Space	15	3		Input

Figure 1: Gen Info output

2 Contribution

Please refer to table 2 for detailed contributions.

3 Related work

The problem statement explored in this project is an ongoing research and competition hosted by The Learning Agency Lab and Vanderbilt University. It's difficult to summarize the complex set of behavioral actions and cognitive activities in the writing process. Writers may use different techniques to plan and revise their work, demonstrate distinct pause patterns, or allocate time strategically throughout the writing process. Many of these small actions may influence writing quality. Even so, most writing assessments focus on only the final product. So, assessing the essay while its being written is mostly unexplored domain and hence there is a dearth of related work. We are trying to uncover this through our project.

4 Methodology

We will be approaching this problem from various perspectives, like Classification, Regression, Deep Learning, etc, and each tailored to extract meaningful insights from the data.. In addressing the classification task, our methodology employs versatile techniques, starting with the Support Vector Classifier (SVC) for its aptness in handling classification tasks. We further try to explore variations such as ordinal classification. Additionally, we delve into deep learning using a Multi-layer Perceptron (MLP) and leverage transformer-based approaches for text data processing.

The tasks will be performed on the two set of data:

- **PCA:** Reduce the dimensionality and improve performance
- **Non PCA:** Allow the use of semantics in data

Contributor	Contribution
Neeraj Gupta	**Focus task:** - Domain level feature engineering - Classification implementation - Regression implementation **Supporting task:** - Mid-term project proposal - Final Report - Presentation - Transformer implementation
Ayush Manojkumar Lodha	**Focus task:** - Ordinal classification implementation - Hyperparameter optimization - Level 1 feature engineering - Data pre-processing **Supporting task:** - Final Report - Mid-term project proposal - Presentation - Deep learning implementation
Dhiraj Pimparkar	**Focus task:** - Sequential feature engineering - Transformer implementation - Deep learning implementation **Supporting task:** - Final Report - Presentation - Classification implementation

Table 2: Contributors and Their Contributions

4.1 Classification

In the classification phase, we employed the Support Vector Classifier (SVC) due to its suitability for classification tasks. The SVC is designed to identify an optimal hyperplane that separates different classes within the feature space. SVC model will be further optimised for its hyperparameters using the validation dataset by hyperopt library in python.

4.2 Ordinal Classification (Suggested during Presentation)

In response to suggestions given during a presentation, we are exploring an ordinal classification approach to better capture the ordinal nature of the scores.

4.3 Regression

For regression analysis, we chose the Light Gradient Boosting (LGB) Regressor known for its efficiency in iterative error correction and regularization. Hyperparameter fine-tuning focused on parameters such as learning_rate, num_leaves, max_depth, subsample, and colsample_bytree.

4.4 Deep Learning

In the deep learning phase, we utilized the Multilayer Perceptron (MLP), a type of feedforward neural network. The model consists of an input layer, an output layer, and 3 hidden layer.

4.5 Transformers

We tackle this challenge by leveraging Transformers, where each word is uniquely represented by an integer index. We implemented the transformer model from scratch due to the uniqueness of the data. To enhance the contextual understanding, we employ the TokenAndPositionEmbedding layer to create embeddings. We set the sequence length to 1000, capturing the initial 1000 events per user per essay. The sequence length was capped at 1000 due to the computational constraints. The subsequent sections delve into the comprehensive discussion of the implementation details and data processing involved in this entire process.

The implementation and the data processing involved in the whole process will be discussed in the following sections.

5 Implementation

A comprehensive preprocessing and data exploration was performed before the implementation of the models.

5.1 Data Preprocessing

In the initial phase of feature engineering, we introduced a variety of features through simple aggregation and determination of maximum values for specific characteristics. These features enhance the representation of user behavior during the essay-writing process. The engineered features include both numerical and categorical types, as detailed in Table 4.

5.1.1 Domain-Specific Feature Engineering

Subsequently, we delved into domain-specific feature engineering, drawing insights from relevant research articles and our understanding of essay writing. The goal was to capture nuances in the dataset,

Feature Name	Feature Type (Categorical/ Numerical)	Description
'id'	Categorical	The unique ID of essay
'event_id'	Categorical	The index of event for user, order chronologically
'down_event'	Categorical	The name of the event when the key/mouse is pressed
'up_event'	Categorical	The name of the event when the key/mouse is released
'down_time'	Numerical	The timestamps of the down event in milliseconds
'up_time'	Numerical	The timestamps of the up event in milliseconds
'action_time'	Numerical	The duration of event (difference between 'down_time' and 'up_time')
'activity'	Categorical	The category of activity which the event belongs to Nonproduction - The event does not alter the text in any way Input - The event adds text to the essay Remove/Cut - The event removes text from the essay Paste - The event changes the text through a paste input Replace - The event replaces a section of text with another string Move From [x1, y1] To [x2, y2] - The event moves a section of text spanning character index x1, y1t o a new location x2, y2
'text_change'	Categorical	The text that changed as a result of the event (if any)
'cursor_position'	Numerical	The character index of the text cursor after the event
'word_count'	Numerical	The word count of the essay after the event
'score'	Numerical – If Model is Regression Categorical – If Model is Classification	Score of the essay from 0 to 6

Table 3: Feature Descriptions

Feature Name	**Feature Type**	**Description**
'down_time_sum'	Numerical	Sum of 'down_time' values for each 'id'.
'up_time_sum'	Numerical	Sum of 'up_time' values for each 'id'.
'action_time_sum'	Numerical	Sum of 'action_time' values for each 'id'.
'cursor_position_max'	Numerical	Maximum 'cursor_position' value for each 'id'.
'word_count_max'	Numerical	Maximum 'word_count' value for each 'id'.

Table 4: Level 1 Feature engineering

resulting in features designed to detect specific occurrences during the essay composition process. The features and their descriptions are outlined in Table 5.

5.1.2 Key Stroke Analysis

To capture the actions undertaken during essay writing, we utilized the 'up_event' column, identifying the types of keys used by the writer, such as 'Enter', 'Space', 'F1', 'F2', etc. We created a user-level event count to quantify these actions.

After the feature engineering process, our dataset transitioned from 8,405,898 instances to 2,471 data points (one per essay/user). The feature set expanded from 11 to 146 features, providing a richer representation of the essay composition process.

5.2 Exploratory Data Analysis

The distribution of essay scores reveals that the majority falls within the 2 to 5 range, as illustrated in Figure 2. This concentration suggests that most essays received scores within this specific range.

A strong correlation is observed between 'Down_time' and 'Up_time,' suggesting potential multicollinearity (Figure 3). The distribution of 'Up_time' and 'Down_time' features also displays noticeable skewness and contains outliers.

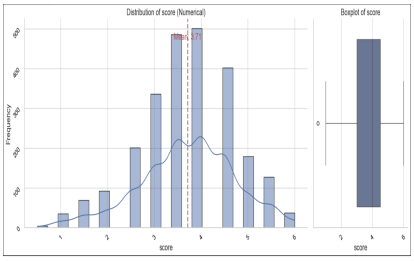

Figure 2: Distribution of Essay Scores

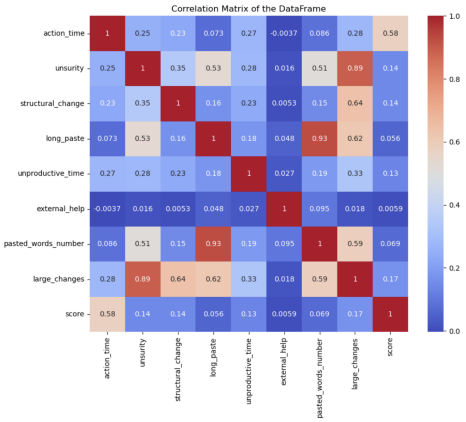

Figure 3: Correlation Heatmap of Features

Feature Name	Feature Type	Description
'unsurity'	Categorical (binary)	Indicates large text removal during 'Remove/Cut' activity.
'structural_change'	Categorical (binary)	Indicates a large text change during 'Replace' activity.
'long_paste'	Categorical (binary)	Flags large text addition during 'Paste' activity.
'unproductive_time'	Numerical	Total time spent in 'Nonproduction' activities.
'external_help'	Categorical (binary)	Indicates reliance on external help during 'Paste' activity.
'pasted_word_numbers'	Numerical	Number of words added during 'Paste' activities.
'large_changes'	Categorical (binary)	Indicates a text change over 50 characters.

Table 5: Domain-Specific Engineered Features

The distribution of scores across users who frequently engage in media actions exhibits an interesting pattern (Figure 4). Users with more media actions tend to achieve scores around the 3.0 mark, suggesting a possible association between media interaction and reduced essay scores.

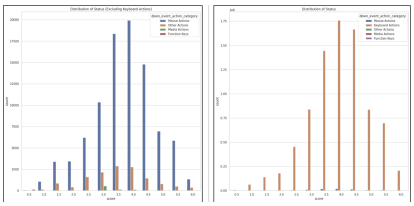

Figure 4: Media Interaction vs. Essay Scores

The ratio of Non-Productive to Remove/Cut actions correlates with essay scores (Figure 5). A decrease in this ratio is associated with higher essay scores, suggesting that minimizing non-productive behavior may enhance performance.

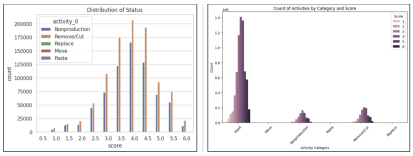

Figure 5: Non-Productive Ratio vs. Essay Scores

Replace, Paste, and Move actions occur less frequently compared to Input and Non-Productive actions during essay writing (Figure 5). Understanding the distribution of these actions is crucial for assessing user behavior during the composition process.

The correlation matrix (Figure 3) suggests that PCA could be beneficial for dimensionality reduction. By applying PCA, we aim to address redundancy and noise in the dataset. The resulting t-SNE plot (Figure 8) displays separate clusters for each score point, revealing the inherent complexity and variance in the dataset.

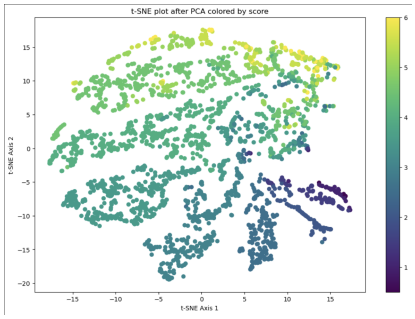

Figure 6: t-SNE Plot of Engineered Features

5.3 MODEL TRAINING AND VALIDATION

We maintained two versions of the data: one with PCA (Principal Component Analysis) applied and the other as non-PCA (original data). By applying PCA, we reduced the number of features to 30, capturing 95% of the dataset's variance. Both versions of the dataset were utilized in all subsequent experiments, including both regression and classification tasks. Find the detailed split between training, validation and testing datasets.

With the SVC, two different kernel settings were tested: one without PCA, where we used a radial basis function (rbf) kernel with C set to 6.9580 and gamma to 0.023, another with PCA, adjusting the SVC parameters according to the hypertuning to C = 0.7204 and gamma = 0.053, still using an rbf kernel.

In case of regression, we fine-tuned the model's hyperparameters, focusing on:

1. `learning_rate`: Controls the learning pace, tested over a logarithmic scale to pinpoint the ideal update rate.

2. `num_leaves`: Governs the number of leaves per tree, influencing model complexity and the risk of overfitting.

3. `max_depth`: Manages tree depth to balance detail capture and the bias-variance trade-off.

	With PCA (95% variance covered)		Without PCA		
	Training	**Validation**	**Training**	**Validation**	**Testing**
X	(2000, 30)	(223, 30)	(2000, 146)	(223, 146)	(248, 146)
Y	(2000, 1)	(223, 1)	(2000, 1)	(223, 1)	(248, 1)

Table 6: Dataset Dimensions with and without PCA

4. `subsample`: Determines the sample portion for fitting base learners, introducing randomness to mitigate overfitting.

5. `colsample_bytree`: Dictates the feature fraction for tree construction, affecting feature diversity in learning.

While the problem was addressed by normal machine learning models and the scope of the course, we wanted to go beyond our comfort zone and test Deep learning and Transformers models. For normal DL, a Multi-layer Perceptron (MLP) is used. It is a type of feedforward neural network composed of three kinds of layers—the input layer, the output layer, and at least one hidden layer. In our case, we implemented three hidden layered MLP. Specifications used for the MLP are

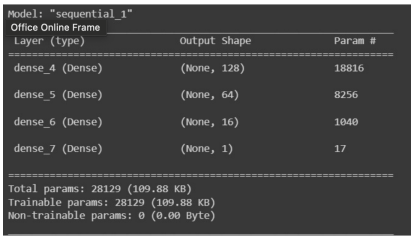

Figure 7: MLP structure

1. **Optimizer:** Adam

2. **Loss Function:** Mean Squared Error

3. **Activation Function:** ReLU

Due to the sequential nature of the dataset, we also tried to model the writing score using the Transformers. The structure of the transformer implemented for this task is,

1. **Integer Vectorization using TextVectorization Layer from Keras:** This step involves representing each word in the input text with a unique integer index, facilitating efficient processing within the neural network.

2. **TokenAndPositionEmbedding Layer:** The TokenAndPositionEmbedding layer plays a crucial role in creating embeddings for the input data. Notably, these embeddings are not explicitly pre-trained on external data. Instead, they are initialized randomly and fine-tuned during the training of the entire model.

3. **Sequence Length: 1000** We consider the first 1000 events per user/essay, ensuring a balanced representation of the input data. This sequence length parameter is chosen to capture sufficient context while managing computational complexity while keeping the computational constraints in mind.

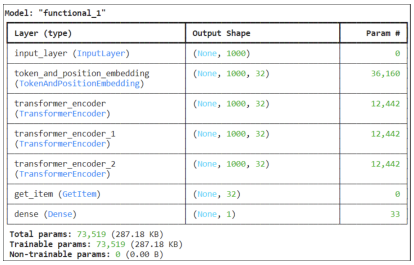

Figure 8: Transformer structure

6 EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Benchmarking

In our proposal, we have identified a range of twelve scores, from 0.5 to 6 in increments of 0.5, to be assigned to users. Our exploratory data analysis (EDA) suggests that the 'score' label for supervised learning is apt for both classification and regression strategies. Given the lack of extensive research on individual benchmarks for classification and regression tasks in this domain, the importance of benchmarking in our project becomes even more critical. By establishing a performance baseline through benchmarking, we can effectively navigate the best approach — whether it's classification, regression, or a hybrid — for accurately predicting these scores.

$$BaselineAccuracy = \frac{NumberofInstanceswithMostFrequentClass}{TotalNumberofInstances} = 0.2027 \quad (1)$$

6.1.1 Classification

To calculate the baseline accuracy for classification tasks, the following formula is used:

The baseline accuracy for classification has been determined to be 0.2027. This will serve as our benchmark for model comparison and drawing conclusions. We will consider a model acceptable if its accuracy surpasses this baseline figure.

6.1.2 Regression

In our regression analysis, we adopted two distinct approaches. Firstly, we posited that a grader assigns scores to each essay by selecting from a normal distribution, yielding an RMSE of 1.4317; this figure serves as our primary baseline for regression models. Secondly, we considered a scenario where each user/essay is given the mean score, resulting in a lower RMSE of 1.0246. The first method reflects grading variability and is more realistic, whereas the second offers a more conservative estimation of grading behavior, each providing valuable perspectives for model evaluation.

6.2 Results

The detailed results are compared in the table 8 and table 7. We tried using an ordinal classification as per the suggestions during the presentation. We explored it using the non-PCA data and we found the results were not promising. Hence we decided not to go beyond that part. We are able to beat the baselines we set at the beginning of our project comfortably.

After going through the results, we observed that both the baseline and the actual results were unsatisfactory hence it makes sense to treat our problem as regression tasks rather than classification.

	With PCA		Without PCA	
	Validation Accuracy	Test Accuracy	Validation Accuracy	Test Accuracy
Normal	0.296	0.2984	0.3094	0.2863
Ordinal	-	-	0.2132	0.2419

Table 7: Detailed results for classification

	With PCA		Without PCA	
Model	Validation error	Test error	Validation error	Test error
Regeression	0.6949	0.7217	0.6248	0.6255
Deep Learning	0.712	0.7651	0.679	0.7045
Transformers	-	-	0.9976	1.1086

Table 8: Detailed results for regression

7 Conclusion

In conclusion, our exploration of machine learning models for both regression and classification tasks has led us to identify `'LGB'` as the optimal choice for regression and `'SVC'` for classification in the context of the competition's specified evaluation metric, `'quality'`, which aligns with the RMSE (Root Mean Square Error).

The decision to prioritize regression is rooted in the competition's focus on `'quality'` as the primary evaluation metric. Since `'quality'` is associated with RMSE, regression becomes the natural choice to optimize for this metric. Additionally, the abundance of target variable `'score'` classes further supports the preference for a regression approach, providing a more nuanced and granular understanding of the predicted scores.

Moving forward, optimization efforts can be extend to the transformer model, pending further exploration and understanding of transformer architectures. This includes fine-tuning the model and addressing potential memory allocation issues on platforms such as Google Colab.

To enhance the model's performance, adjustments to the sequence length are necessary. Resolving memory allocation issues on Google Colab will facilitate the expansion of the sequence length, allowing for a more comprehensive analysis of essay data.

Furthermore, future work can involve advanced feature engineering based on recommendations from domain experts. Implementing more sophisticated features tailored to the nuances of essay writing will likely contribute to the model's predictive capabilities.

In summary, our conclusion not only identifies the best-performing models for the given tasks but also outlines a roadmap for future improvements, encompassing transformer optimization, sequence length adjustment, and advanced feature engineering. This strategic approach aims to elevate the model's accuracy and effectiveness in predicting essay scores in everyday use.

References

[Linking Writing Processes to Writing Quality 2023] The Learning Agency, Vanderbilt University 2023. https://www.kaggle.com/competitions/linking-writing-processes-to-writing-quality/overview In *Kaggle*

8 Appendix

Group members in order: Ayush Manojkumar Lodha, Neeraj Gupta, Dhiraj Pimparkar