JOURNAL OF
EMERGING INVESTIGATORS

Article

Gradient boosting with temporal feature extraction for modeling keystroke log data

Ryan Barretto[1], **Sanjay Barretto**[1]
[1] William Fremd High School, Inverness, Illinois

SUMMARY

Although there has been great progress in the field of Natural language processing (NLP) over the last few years, particularly with the development of attention-based models, less research has contributed towards modeling keystroke log data. State of the art methods handle textual data directly and while this has produced excellent results, the time complexity and resource usage are quite high for such methods. Additionally, these methods fail to incorporate the actual writing process when assessing text and instead solely focus on the content. Therefore, we proposed a framework for modeling textual data using keystroke-based features. Such methods pay attention to how a document or response was written, rather than the final text that was produced. These features are vastly different from the kind of features extracted from raw text but reveal information that is otherwise hidden. We hypothesized that pairing efficient machine learning techniques with keystroke log information should produce results comparable to transformer techniques, models which pay more or less attention to the different components of a text sequence in a far quicker time. Transformer-based methods dominate the field of NLP currently due to the strong understanding they display of natural language. We showed that models trained on keystroke log data are capable of effectively evaluating the quality of writing and do it in a significantly shorter amount of time compared to traditional methods. This is significant as it provides a necessary fast and cheap alternative to increasingly larger and slower LLMs.

INTRODUCTION

Transformer attention-based methods have largely dominated text-based supervised learning over the last few years, mainly due to publicly available large language models (LLMs) (1, 2). These models allow for highly efficient and effective fine tuning on downstream tasks, such as sequence classification and question answering. Despite the reduced time it takes to fine tune LLMs relative to training a model entirely from scratch, the training process can still take considerable time and requires support from a graphics process unit (GPU).

Bidirectional encoder representations from transformers (BERT) were one of the first LLMs and also one of the first models to utilize attention, which gave it the unique ability of understanding how words relate to one another. Decoding-enhanced BERT with disentangled attention (DeBERTa), an LLM that was built on BERT, is currently ranked seventh for the General Language Understanding Evaluation (GLUE) benchmark (3,4). The GLUE benchmark is a large dataset that creates several fundamental tasks of natural language processing (NLP) including question answering and text classification, amongst others. The base version of the BERT model comprises 86 million total parameters while the large version has over 300 million parameters. To be able to fine-tune such a model on a task, a GPU and a considerable amount of time is required. Additionally, achieving optimal performance requires hyper parameter tweaking, such as training for more/less epochs (training iterations) and increasing/decreasing the learning rate (how aggressive the training approach is), which further the total time complexity. Traditional methods attempt to use classical machine learning algorithms (as opposed to deep learning neural networks) paired with text feature extraction tools, such as term frequency – inverse document frequency vectorizers (TF–IDF) to model textual data (5). These methods are computationally simpler than current state of the art techniques, but do not achieve the same level of performance. The primary issue with such methods is that they fail to incorporate textual relationships into the features they provide. However, these methods do indicate that machine learning methods can be viable so long as meaningful features are provided. Methods that utilize information from the writing process, rather than just information from the produced text itself, do exist and have achieved success. Keystroke logging is the act of recording keyboard events while a user is typing. The data collected from keystroke loggers can be very informative and reveal information about textual data that the text itself cannot (6). For example, suppose a writer takes a deep pause when writing, perhaps planning out the rest of their essay or for some other deliberation. Looking at just the written text, it would be impossible to assess that such a pause was ever taken. However, with keystroke log data, one can analyze the time between clicked keys and discover such events.

Due to the differences in the way current LLM methods and keystroke log methods approach text-based problems, a hybrid approach could improve the results of both. LLM methods offer a detailed look into the text itself and the deeply encoded properties within it, while the keystroke log approach uncovers details about the writing process, missing from the final product. A combination of both should therefore provide the best results on downstream tasks. However, if resources are limited, an approach solely using keystroke log data could be advisable as one can fully train a model and perform inference with just a CPU in a very short amount of time. Therefore, the use of keystroke log data could not only serve as a source of improvement on NLP performance but could also serve as a speedy alternative in many situations.

https://doi.org/10.59720/24-087

Prior research has been conducted on utilizing keystroke log data for modeling certain downstream tasks. Positive relationships have been found between keystroke information and students' writing processes by modeling keystroke log data with heavy tailed probability distributions (7). Specifically, work has demonstrated that task engagement and writing efforts may play a big role in the overall quality of writing produced (8). Other research focused on assessing student writing quality by modeling features extracted from keystroke log data (9). This work demonstrated promising results for uncovering relationships between keystroke log data and writing quality. Each of these works suggest that keystroke information may have strong predictive power for the quality of writing, However, due to the older models that were used in these papers, we believe that the results could be improved upon with newer, advanced machine learning models. Research has also shown that just the number of words written in an essay within the first 999 keystrokes could be powerful predictors for machine learning models (10). On a similar note, work has shown writing quality could be modeled successfully with just two to five features (variables that can be manipulated to make predictions), with the number of keystrokes for a given essay being the most important (11). Research in 2022 looked at identifying students in need of assistance using a variety of features similar to those extracted from the 2022 study, using models such as Naive Bayes, SVM's, and random forests (12). While this research was insightful, the methods did not address assessing the quality of writing.

Recently, research has been conducted examining the use of keystroke dynamics paired with machine learning models (13). In this research, the authors use classical machine learning models like SVM's, random forests, and K-nearest neighbors, in addition to deep learning methods like neural networks. These models were used to analyze keystroke log data to distinguish between the different users who typed responses. The authors ultimately offered a comparative analysis of different models for biometric identification (determining the identity of a person), while we have focused on a single gradient booster's performance on keystroke log data for analyzing the quality of students' writing versus the performance of LLMs on the same task.

We hypothesized that pairing efficient machine learning techniques with keystroke log information will produce results comparable to transformer techniques in less time. We demonstrated that gradient boosters, machine learning algorithms that merge the ideas of neural networks and random forests into one algorithm, using keystroke information are able to produce strong results on assessing student writing in under a second. It is clear that keystroke log data, normally consisting of keypress speed and frequency, which describes the writing process, can be just as useful for understanding student writing as the actual writing itself. Our results suggest the writing process reveals important information that is often missed in traditional NLP techniques, specifically within the task of evaluating the quality of writing. This additional information could be helpful not only by itself for efficient and fast data modeling as shown in this paper, but also paired with the original text may provide the entire picture of a student's writing which could lead to improvements on modeling in the field as a whole.

RESULTS

Raw keystroke data consists of the type of key event taking place over time and is then grouped according to the session it took place. Features that were extracted from this raw data involve the number of keystroke events, average time taken per keystroke and more (**Table 1**). A gradient boosting classifier was then trained on this extracted data to distinguish between different qualities of writing. This particular classifier was chosen because it has been shown to be extremely efficient and accurate when dealing with tabular data. The main metrics for evaluating the model are the root mean square error (RMSE) and quadratic weighted kappa (QWK).

The major results looked at are the times it took for a certain model to train and the scores it achieved in predicting the quality of an essay given the actual quality it received by human raters. On average, each model scored an RMSE of 0.6624 taking a total of 0.3774 seconds to train **(Figure 1)**. The normalized RMSE for each model, on average, was 0.1104, due to the range of our target variable being equal

Created Feature	Description	Calculated
Verbosity	Number of keystroke events per user response	Count the number of occurrences of a specific **ID**
Backspaces	Number of times the backspace key was pressed during writing	Count the number of times per user that **DownEvent** equaled Backspace
WordCount	Final number of words in the response	Find the last **WordCount** chronologically
SentCount	Final number of sentences in the response	Count the total number of periods occurring within the essay
ParagraphCount	Final number of paragraphs in the response	Count the total number of times the ENTER key was pressed
Nonproduction	Total number of key presses that didn't contribute a change to the response	Count the number of times
AvgKeystrokeSpeed	How many seconds, on average, it took a user to press down on a key	Divide **Verbosity** by the total elapsed time.
LargestInsert	The largest difference in word counts between subsequent events	Calculate differences in **WordCount** between back to back keystroke events and save the max.
LargestDelete	The smallest difference in word counts between subsequent events	Calculate differences in **WordCount** between back to back keystroke events and save the min.
LargestLatency	The maximum amount of time spent not making changes to the writing	Largest difference in **DownTime** between subsequent keystroke events.
SmallestLatency	The smallest amount of time spent not making changes to the writing	Smallest difference in **DownTime** between subsequent keystroke events.
MedianLatency	The median amount of time spent between keystroke events	Sort differences between subsequent keystrokes and find the middle value.
FirstPause	First pause taken during the writing process	Initial difference between two keystrokes.
Pause (0.5, 1, 1.5, 2, 3)	the number of (0.5, 1, 1.5, 2, 3) second pauses taken during writing	Count the number of second differences between consecutive keystrokes that are in the range of (0-0.5, 0.5-1, 1-1.5, 1.5-2, 2-3).

Table 1: Base-level extracted features synthesized from general patterns discovered within the raw keystroke log data. These features represent simple patterns that were recovered from an individual's writing process. Most features relate to the time it took for keystroke events as well as the types of keystrokes occurring.

https://doi.org/10.59720/24-087

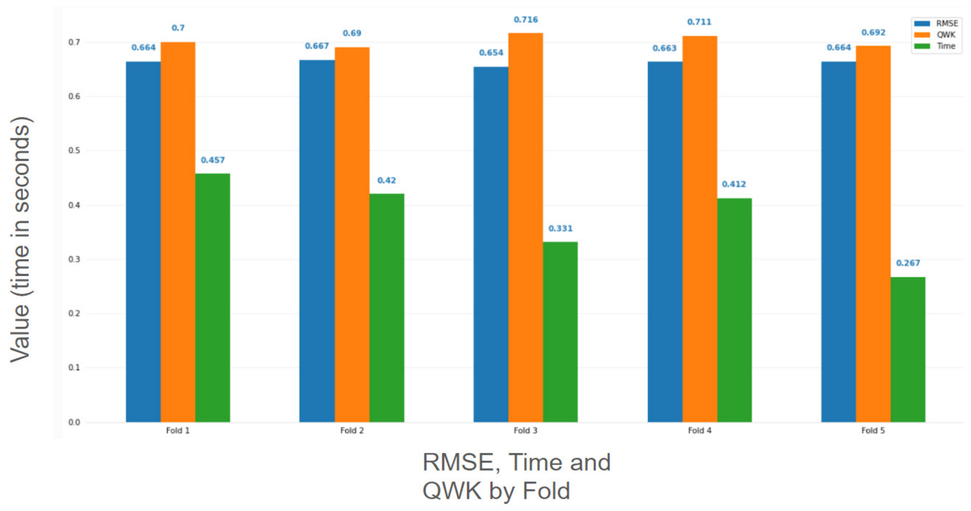

Figure 1: LightGBM performance on evaluating student essays based on RMSE and QWK. This graph shows the results the model obtained for each of the 5 data splits, including its quantitative performance and the time it took to train. An average RMSE of 0.6624 was obtained taking 0.3774 seconds on average.

to a constant six. We used LightGBM to model the data, a powerful gradient booster that combines accuracy with efficiency. Additionally, the hierarchical-based processing of the LightGBM model allows for feature importance to be estimated (14). The relative importance of each feature was calculated with word count, paragraph count, and pause frequencies being among the most important **(Figure 2)**. We also graphed and analyzed the types of error **(Figure 3)**. The model was fairly even in the errors it made, overestimating and underestimating the score an essay received fairly equally. Additionally, barring any outliers, the model was at most off by two points for a grade when assessing a given essay, indicating that there was generally a strong agreement between the predictions and the official scores.

To fully understand how the performance of this model compares to LLM based methods and to address our original hypothesis, we compared our results to prior research that used LLMs to assess the quality of essay writing (16). We specifically examine Table 3 of the paper as it features the QWK scores for two tasks most related to our own (automatic essay grading with a score range of 1–6). Our model scores an average QWK of 0.7018 **(Figure 4)**. In comparison, most of the LLMs featured in the paper achieve under 0.7 for the first experiment and above 0.775 for the second experiment.

DISCUSSION

In this paper, we highlight a different way of approaching natural language tasks, one that doesn't involve the language

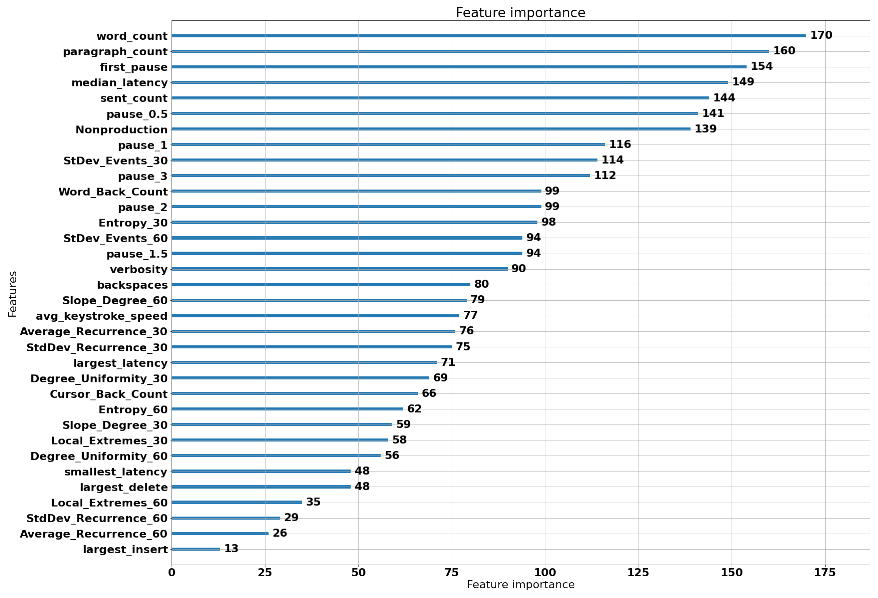

Figure 2: LightGBM's feature importance. The importance of each feature to the model trained on a specific data split, where a higher numerical value means that a feature is relatively more important than another. This means that a feature. The number of words and paragraphs are shown to be most important to the model.

https://doi.org/10.59720/24-087

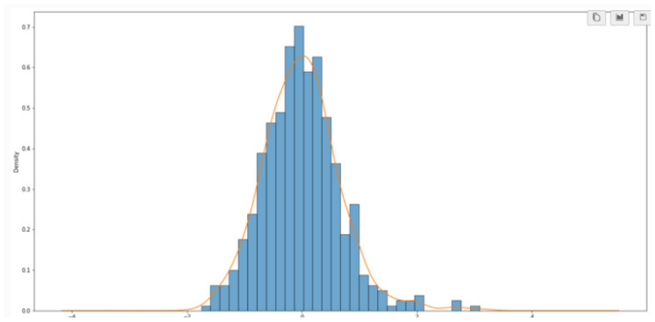

Figure 3: Prediction residuals. The above figure shows a histogram of the differences between the predictions made and the true targets.

itself. In the last few years, attention-based methods have taken over the field of NLP. While these methods have drastically improved on the results obtained by older methods, they have also increased the run time and resources required to model such data. Right now, the norm appears to be fine-tuning models with billions of parameters to achieve strong results on down-stream tasks. Specifically, the transformers we compared to were trained with a Nvidia RTX8000 GPU and a majority of the models used had well over a million parameters. These results reveal two very important comparisons with our own experiment. Firstly, our own gradient booster is able to achieve a very similar performance on scoring essays as LLMs, as can be seen with the numbers above. Secondly, we can see that without the use of a GPU, our own model requires significantly less time or computing power due to the significantly smaller nature of the model. This supports our original hypothesis as our gradient booster using keystroke features performs at a level comparable to transformer based LLMs while using less time and computing power.

Keystroke log data also gives us insight that we are not able to observe normally with raw textual data. Although incorporating keystroke information with the actual text data could further improve results, we found only using the keystroke log data was not only effective but also highly efficient. Gradient boosters are extremely quick and precise when handling datasets with a relatively low number of features (around 40 used here). Additionally, the writing process itself offers a new way to view and process textual data. Rather than the words that ended up on the final paper, the words that were deleted, sentences that were edited, and other features that are missing from the final product could be used as inputs to simpler machine learning models. Machine learning models have been used in the past for processing text, typically paired with embedding models like term frequency-inverse document frequency (TFIDF) vectorizers, GLOVE vectors, and FastText amongst many others (17, 18). Although these methods are often more cost-effective than transformer-based methods, they usually perform worse. Additionally, with almost all of these methods, the textual data is transformed into a feature space with a high dimensionality, resulting in a time-consuming training time even for smaller machine learning models.

However, using gradient boosters with keystroke log data, we were able to exploit the writing process to represent an essay with a significantly smaller feature set. Additionally, a strong normalized RMSE demonstrates the strong capability of gradient boosters trained on keystroke log features. Further task-related optimization could drop this loss further, such as model fine-tuning or more feature engineering.

There were some limitations and shortcomings to note while interpreting these results. The first major limitation regards the process of collecting keystroke log data. Unlike raw textual data, which can easily be recorded in the form of a response or scraped off the internet, keystroke log data requires software to collect the data and permissions from users. This is a far more complicated process for data collection compared to traditional methods, which could make it difficult to use keystroke log data in certain tasks. Additionally, performing inference with keystroke log data would require keystroke log software to be active while a new user is typing. The data collection phase is more difficult for keystroke log data compared to raw textual data and limits the usability of the methods provided here.

Another limitation faced in this paper is a lack of reliable comparison between keystroke log methods and state of the art transformer-based methods. Due to the anonymous

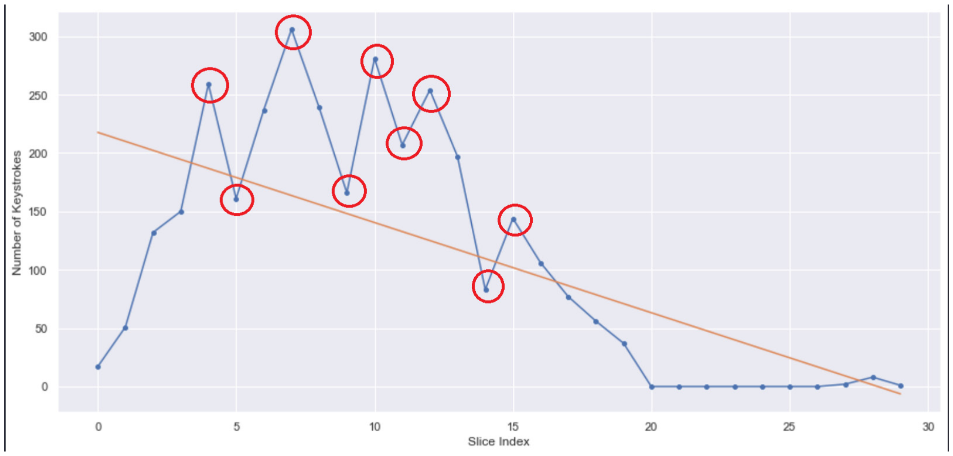

Figure 4: Sample time series displaying the number of keystrokes over time. The number of keystrokes in each 60 second time slice (sequential segments of 60 second time spans where an index corresponds to the position of the segment) for a single participant while they were writing (blue). It also shows the line of best fit used to determine slope degree (for 30 and 60 second time slices) (orange) and some noticeable local maxes and mins (red).

https://doi.org/10.59720/24-087

nature of the data used in this paper, it is not possible to see how a transformer model would have scored on the original text data nor is it possible to see the performance of a hybrid model. While we did attempt to compare the performance of LLMs compared to gradient boosters, this comparison was not made on the same set of data, so it instead serves as a rough estimate rather than a completely precise measure.

Using keystroke log data to model textual inputs is a less explored part of NLP and we presented ideas intended to serve as a baseline. Several changes or additions could improve the results. These revisions can be as simple as optimizing the hyper-parameters of the model and ensembling or stacking several models, or as complicated as engineering new features from the time-series component of the data. Another important development could be the creation of a non-anonymized keystroke log dataset. This would allow researchers the ability to compare the strength of models using keystroke log data to the strength of models using textual data. This could show the validity of keystroke-based methods as well as highlighting the trade-off in efficiency versus performance. Additionally, this data could be used to explore hybrid methods which utilize both the final product and the process of getting to the final product. Such a model would be able to get a strong variety of features, which could allow for even stronger results. Another potentially interesting form of research with keystroke log data is a longitudinal analysis. Uncovering changes in the way people type as a function of time could be both interesting and insightful.

While we focused on optimizing time complexity, optimizing performance could also be achieved. The easiest and possibly most effective way for achieving stronger results is to adjust the parameters of the model. For example, increasing the number of individual estimators in the model could boost the performance of a model, albeit while increasing the training time. Other parameters, like the learning rate or number of leaves can be specifically obtained through techniques like Bayesian optimization (15). While Bayesian optimization is a certain way of improving the results of the booster, the process of recovering parameters that yield significant improvements can be time-consuming, especially if we have a larger number of data points. Therefore, employing these improvements should be considered if the user is willing to make a time versus performance sacrifice.

In addition to assessing writing quality, it is possible to recover the overall structure of the essay using the keystroke log information. For example, one can reconstruct the final finished essay just with anatomized characters (all characters replaced with a constant character). Of course, this new essay will not carry any of the actual content as the original, but it could be processed by a transformer model to reveal deeper structural features, as transformer based models can extract information from the structure of a text. Pairing this with additional features could improve the model results. Once again, however, this would result in an increased training time and would likely need the support of a GPU. A GPU is a processor specialized for performing complex mathematical operations, similar to those seen when training large neural networks like transformers. However, these processors are often expensive and not readily available. Rather than improving the model itself, we believe that improvements to the data could be both more efficient and more effective. Crafting new features would not drive up the training time significantly but could lead to improvements in performance.

To summarize, we introduced a new framework for handing keystroke log data and a different approach for handing textual inputs. Many existing LLM models require hours to train precisely. Our results highlighted a significantly shorter training time for our models and an average normalized RMSE across five folds of 0.1104, indicating the reliability of using gradient boosters to assess the quality of text. Additionally, we demonstrated that gradient boosters using keystroke log data achieved comparable results to LLMs on similar tasks with QWK scores of roughly 0.7 amongst all models. Furthermore, the similar performance of the model proposed in this paper was reached in significantly less time and with significantly less resources. These results directly address and support our original hypothesis. However, future work must be done to reveal the potential of hybrid models which capture both the writing and the writing process.

MATERIALS AND METHODS

A Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz processor was used with 16 GB of memory. Additionally, all scripts were run in python on VSCode. The model we employed was a LightGBM regressor, which utilizes decision trees as its underlying model (14). Decision trees work by continuously partitioning data based on conditions and then assigning values to partitioned groups. One such example of this is "if age is greater than 21, classify as adult, else classify as child." LightGBM combines hundreds of these decision trees while also optimizing the size of the trees. In this paper, the LightGBM model had 100 decision trees and was trained until the loss (a measure of how well the model could predict new essays) stopped decreasing.

Each response was assigned one score that will be predicted, but each response has over 3000 individual keystroke events that were made (19). This means that we have to develop features that either aggregates keystroke events into one cumulative feature or incorporates information on the time series aspect of a group of keystroke events. Let us imagine that we have N responses, each with M keystroke events and each keystroke event carries L features. The dimensionality of this data is (M * N) * L, which does not fit our target dimensionality of N. In other words, we do not have a single target for a single row of features. By performing feature extraction to this raw data, we are able to transform the raw data into N * L dimensionality which is suitable for our machine learning models as we can now use L features to predict a single output variable. Such features mostly involve mean, min, max, and other calculations which pick up on higher level patterns in the data **(Table 1)**. Additionally, many features in this paper come from or are built upon features created in previous papers (8,9,11).

The next set of features that were extracted were based on the time series component of the data. 30 and 60 second slices were created from the raw data and the number of events that occurred within each interval were recorded **(Figure 4)**. For a particular feature, 30 indicates time slices of length 30 seconds and 60 indicates time slices of length 60. The main goal in extracting features from the time series aspect of writing is to uncover patterns like a writer's consistency, focus, organization and other features that can't be picked up from simple statistics or the actual writing itself **(Table 2).**

Created Feature	Description	Calculated
Slope Degree (30, 60)	The approximate slope of Figure 1	A line of best fit is fitted to the time series windows and the coefficient is recovered.
Entropy (30, 60)	Shannon entropy is calculated for a time series like the one shown in figure 1.	The Shannon entropy equation is used for this calculation where i is the time slice index and x is the number of keystrokes in that particular slice. $H(x) = -\sum_i P(x_i) log(P(x_i))$
Shannon Jensen Divergence (30, 60)	The Shannon Jensen Divergence is calculated for a time series like that in figure 1 which measures the distance between a uniform keystroke distribution and the distribution that occurred	The Shannon Jensen divergence is applied to a modified version of the time series where P(i) is the probability that a keystroke occurred in the *i*th time slice and Q is a uniform probability distribution. $KL(P\|\|Q) = \sum_i p(i) * log(p(i)/q(i))$ $JSD(P\|\|Q) = (KL(P\|\|M) + KL(Q\|\|M))/2$
Local Extremes (30, 60)	The number of time slices where the graph changes from increasing to decreasing or vice versa.	Differences between time slice values were calculated and then consecutive differences with opposite signs were tallied.
Average Recurrence	The average of lengths of sequences of time slices with at least one keystroke is taken.	Stdev Recurrence & The standard deviation of lengths of sequences of time slices with at least one keystroke is taken.
AR (Collected on 30 second and 60 second time slices)	The parameters of an autoregressive model fit to the time series like seen in figure 1	
Stdev Recurrence	The standard deviation of lengths of sequences of time slices with at least one keystroke is taken.	

Table 2: Time series extracted features derived from a keystroke vs time based analysis. These features represent information that was extracted from a time series formed from the number of keystrokes an individual typed over time.

To evaluate the model, we chose RMSE and QWK. In the RMSE equation (Equation 1), i denotes the *i*th sample in the set, y represents the sequence of targets we are predicting, ŷ represents the corresponding sequence of predictions our model made, and N is the total number of samples there are. Additionally, y(i) is the *i*th sample in the sequence. In this paper, y represents the target variable which is the scores of essays determined by human raters taking on values from 0.5 to 6. In the QWK equation (Equation 2), i and j denote the *i*th row and *j*th column of a matrix. w represents a weight matrix, O represents a confusion matrix, and E represents the expected value matrix.

Equation 1

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Equation 2

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

All code used in this work for generating the models and analyzing data can be found within the following github repository: https://github.com/Ryan6407/Modeling_KeystrokeLogData/tree/main.

Received: March 9, 2024
Accepted: June 18, 2024
Published: October 4, 2024

REFERENCES

1. Vaswani, Ashish, et al. “Attention Is All You Need.” *ArXiv.org*, 5 Dec. 2017, https://doi.org/10.48550/arXiv.1706.03762.
2. Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *Proceedings of the 2019 Conference of the North*, vol. 1, 2019, https://doi.org/10.18653/v1/n19-1423.
3. He, Pengcheng, et al. DeBERTa: Decoding-Enhanced BERT with Disentangled Attention. 5 June 2020, https://doi.org/10.48550/arxiv.2006.03654.
4. Wang, Alex, et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” *ArXiv*, 20 Apr. 2018, https://doi.org/10.48550/arxiv.1804.07461.
5. Das, Bijoyan, and Sarit Chakraborty. “An Improved Text Sentiment Classification Model Using TF-IDF and next Word Negation.” *ArXiv (Cornell University)*, 17 June 2018, https://doi.org/10.48550/arxiv.1806.06407.
6. Olzak, Tom. “Keystroke logging (keylogging).” Adventures in Security, 8 April 2008, pp. 1-6.
7. Guo, Hongwen, et al. “Modeling Basic Writing Processes from Keystroke Logs.” *Journal of Educational Measurement*, vol. 55, no. 2, June 2018, pp. 194–216, https://doi.org/10.1111/jedm.12172.
8. Sinharay, Sandip, et al. “Prediction of Essay Scores from Writing Process and Product Features Using Data Mining

https://doi.org/10.59720/24-087

Methods." *Applied Measurement in Education*, vol. 32, no. 2, 13 Mar. 2019, pp. 116–137, https://doi.org/10.1080/08957347.2019.1577245.

9. Allen, Laura K., et al. "{ENTER} ing the Time Series {SPACE}: Uncovering the Writing Process through Keystroke Analyses." *International Educational Data Mining Society*, 29 Jul. 2016, pp. 22-29.
10. Likens, Aaron D., Laura K. Allen, and Danielle S. McNamara. "Keystroke Dynamics Predict Essay Quality." *CogSci 2017 - Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, 26 Jul. 2017, pp. 2573-2578
11. Choi, Ikkyu, and Paul Deane. "Evaluating Writing Process Features in an Adult EFL Writing Assessment Context: A Keystroke Logging Study." *Language Assessment Quarterly*, 18 Sept. 2020, pp. 1–26, https://doi.org/10.1080/15434303.2020.1804913.
12. Talebinamvar, M., Zarrabi, F. "Clustering students' writing behaviors using keystroke logging: a learning analytic approach in EFL writing." *Lang Test Asia,* 7 Feb. 2022, https://doi.org/10.1186/s40468-021-00150-5.
13. Chang, Han-Chih, et al. "Machine learning and deep learning for fixed-text keystroke dynamics." *Artificial Intelligence for Cybersecurity. Cham: Springer International Publishing*, 1 Jul. 2021, pp. 309–329, https://doi.org/10.48550/arXiv.2107.00507.
14. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems 30*, 4 Dec. 2017, pp. 3149–3157.
15. Akiba, Takuya, et al. "Optuna: A Next-Generation Hyperparameter Optimization Framework." *ArXiv (Cornell University)*, 25 July 2019, https://doi.org/10.48550/arxiv.1907.10902.
16. Ormerod, Christopher M., Akanksha Malhotra, and Amir Jafari. "Automated essay scoring using efficient transformer-based language models." *arXiv preprint arXiv:2102.13136*, 25 Feb 2021, https://doi.org/10.48550/arXiv.2102.13136.
17. Pennington, Jeffrey, et al. "Glove: Global Vectors for Word Representation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543, https://doi.org/10.3115/v1/d14-1162.
18. Bojanowski, Piotr, et al. "Enriching Word Vectors with Subword Information." *ArXiv.org*, 19 June 2017, https://doi.org/10.48550/arXiv.1607.04606.
19. Alex Franklin, Jules King, Maggie Demkin, Perpetual Baffour, Ryan Holbrook, Scott Crossley. "Linking Writing Processes to Writing Quality". *Kaggle*, 2023, https://kaggle.com/competitions/linking-writing-processes-to-writing-quality.