# Modeling Logit Distributions from Adversarial Large Language Models for AI-generated Text Detection

Dallin Gordon dgordon@bu.edu
GM Harshvardhan gmharsh@bu.edu
Himanshu Patil hipatil@bu.edu

## Abstract

*The rapid advancement of large language models (LLMs) has raised significant concerns in the educational sector, particularly regarding their potential to enable plagiarism and impact students' skill development. This paper addresses the challenge of distinguishing essays written by middle and high school students from those generated by various LLMs. We present a novel machine learning model designed to identify subtle differences between human-written and AI-generated texts. The model is trained and evaluated on a dataset comprising a mix of student essays and outputs from diverse LLMs, and is highly intelligible at the word-level. Our approach focuses on detecting unique linguistic and stylistic features characteristic of LLMs, thereby identifying potential AI artifacts in the text. By analyzing essays on a range of subjects and lengths typical of a classroom setting, our model aims to provide a tool for educators to safeguard academic integrity and assess student work accurately. This study not only contributes to the growing field of AI text detection but also explores the implications of LLMs in educational settings, fostering a balance between leveraging AI as a learning aid and maintaining essential writing skills development in students. All code can be found at* `github.com/HiPatil/EssayAuthenticatorAI`.

## 1. Method

Consider a corpus of documents $\boldsymbol{d}$, where each document $\boldsymbol{d^{(i)}}$ comprises a multiset of $n$ tokens $\boldsymbol{T_{d^{(i)}}} = \{t_0, t_1, ..., t_n\}$, a pretrained LLM $f(\boldsymbol{x}; k)$ which accepts a window of tokens $\boldsymbol{x}$ of length $k$, and outputs a logit distribution with softmax probabilities for the next token $\hat{t}_{k+1}$. The cardinality of the logit distribution of $f$ is equal to the size of the token vocabulary $V$, as it assigns a probability $P(t_j)$ for each token $t_j \in V$. Suppose we have an ordered distribution over the next token given by eq. 1,

$$f(\boldsymbol{x}; k) = \{P(t_0), P(t_1), ..., P(t_{|V|})\} \tag{1}$$

where $P(t_0) \leq P(t_1) \leq ... \leq P(t_{|V|})$, and $\sum_j P(t_j) = 1$. In a typical text generation setting, we obtain the next token as $\hat{t}_{k+1} \leftarrow V[\arg\max_j f(\boldsymbol{x}; k)]$, however, we are interested in finding the index of the ground-truth next token $t_{k+1}$ in the logit distribution of $f$, which can be denoted as $j_{t_{k+1}}$. Repeating this process by sliding the window $\boldsymbol{x}$ forward across all tokens in $\boldsymbol{T_{d^{(i)}}}$, we obtain $(n-k+1)$ indices (or ranks) for the ground-truth next tokens in the logit distributions at each step[1]. This process is visually shown in Figure 2 and analytically in Algorithm 1. Thus, we obtain a rank sequence $\boldsymbol{R^{(i)}}$ associated with every document $\boldsymbol{d^{(i)}}$.

**Algorithm 1** Rank sequence generation

**Input:** $\boldsymbol{d}$, $f(\boldsymbol{x}; k)$
**Output:** $\boldsymbol{R} \; \forall \; \boldsymbol{d^{(i)}} \in \boldsymbol{d}$

$\boldsymbol{R} \leftarrow \{\}$
**for** $\boldsymbol{d^{(i)}} \in \boldsymbol{d}$ **do**
  $\boldsymbol{R^{(i)}} \leftarrow \{\}$
  **for** each $k$-sized window $\boldsymbol{x}$ in $\boldsymbol{T_{d^{(i)}}}$ **do**
    **compute** $f(\boldsymbol{x}; k)$
    $j_{t_{k+1}} \leftarrow j$, where $f(\boldsymbol{x}; k)[j] = t_{k+1}$
    **add** $j_{t_{k+1}}$ to $\boldsymbol{R^{(i)}}$
  **end for**
  **add** $\boldsymbol{R^{(i)}}$ to $\boldsymbol{R}$
**end for**

For our problem, we have two types of documents: $D_{fake}$ and $D_{real}$ within $\boldsymbol{d}$. We store the generated rank sequences separately based on the binary label of the document. In Figure 1 these generated rank sequences are represented as probability distributions $P(D_{fake})$ and $P(D_{real})$ for AI-generated and real documents, respectively. These probability distributions can be obtained by normalizing the frequency distributions of the rank sequences $R_{fake}$ and $R_{real}$. The underlying principle of our approach is that $P(D_{fake})$ would be far from $P(D_{real})$, thus making discriminating among them easy via distance-based or

[1]The window has a growing-phase which starts from size 1 till $k$ before shifting across to obtain $(n-k+1)$ additional indices. For simplicity, the growing phase is neither shown in Algorithm 1 nor Figure 2.

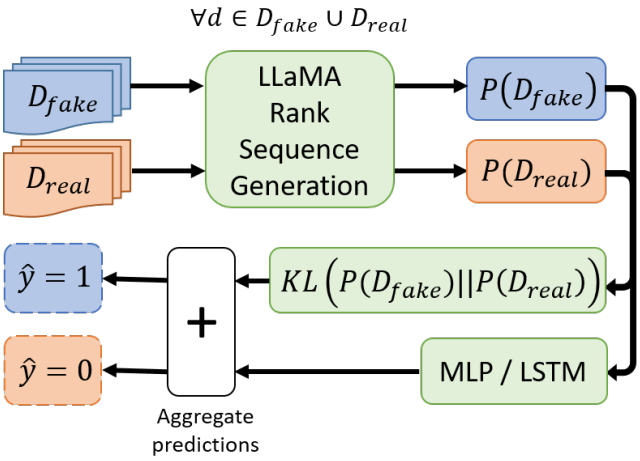


Figure 1. General workflow of proposed method. Sequences from real and AI-generated documents are passed in a sliding window through LLaMA-7B (see Figure 2) to obtain rank sequence distributions $P(D_{fake})$ and $P(D_{real})$ for AI-generated and real documents $d$, respectively. These distributions are used to discriminate samples during inference through KLD and learnable modeling.

learning-based methods. The distance-based approach we use involves the Kullback-Leibler Divergence (KLD) [1–3] defined by eq. 2 for measuring the distance between two probability distributions $p$ and $q$.

$$D_{KL}(p||q) = \sum_i P(i) \cdot \log\left(\frac{P(i)}{Q(i)}\right) \tag{2}$$

One problem with the KLD formulation is that $D_{KL}(p||q) \neq D_{KL}(q||p)$. This could create different results depending on the directionality of distance measurement between real and fake documents. Thus, we allow for a symmetric calculation as shown in eq. 3.

$$KLD(p,q) = \frac{D_{KL}(p||q) + D_{KL}(q||p)}{2} \tag{3}$$

This symmetric formulation violates the triangle property between three or more probability distributions, but we ignore it as it is trivial to our treatment of KLD for two distributions. Initially, we iterated over all documents to make an average label distribution for real and fake documents, however, the averaging rendered the KLD between them to collapse. Thus, currently, we only obtain $P(D_{fake})$ from a few AI-generated samples and during inference, we measure the KLD between the test sample rank sequences (after normalizing their frequency distribution) with $P(D_{fake})$. If the distance is over a threshold $\tau$, we classify the sample as real, and vice-versa if below the threshold.

We also make use neural networks that can learn from the distributions. More to be added.

### 1.1. Dataset

Being a part of Kaggle competition, this project includes dataset with around 10,000 essays. This dataset is uniquely composed of two types of essays: those written by middle and high school students and those generated by a variety of large language models (LLMs). The primary objective of using this dataset is to develop a machine learning model capable of distinguishing between human-written and LLM-generated essays.

The essays in the dataset were all crafted in response to one of seven carefully chosen essay prompts. These prompts required the writers to read and comprehend one or more source texts before writing their essays. In the case of LLM-generated essays, this source material was sometimes, but not always, provided as input to the LLMs.

Our dataset is divided into two main subsets: the training set and the hidden test set. The training set consists of essays responding to two of the seven prompts. This set is predominantly comprised of student-written essays of around 1300 essays, supplemented by 700 essays of LLM-generated essays to serve as examples. To enhance the robustness of our training process, we generated of additional essays using various GPT-3.5 and GPT-4, ensuring a balanced and comprehensive training set.

The hidden test set, designed to evaluate the model's efficacy, includes essays from the remaining five prompts. This subset is undisclosed to prevent overfitting and to ensure that our model's performance is evaluated under realistic and unbiased conditions.

Our approach in structuring the dataset is intended to mirror real-world scenarios where educators and researchers might need to identify the origin of a text. By encompassing a diverse range of subjects, writing styles, and lengths typical in an academic environment, we aim to develop a model that is both accurate and generalizable across different contexts and LLMs.

### 1.2. Preprocessing

Each token in LLM generated text is sampled from a distribution. Since LLMs are trained on colossal corpora, word choice idiosyncrasies are averaged out. Our theory is that essays generated by individuals would have more statistically surprising text than one wholly generated by an LLM. To quatify the surprisingness of a word choise, we decided to use an LLM. Llama [4], Meta's 7 billion parameter open source LLM, was the model we chose to use. The inference method takes a string of characters that have been tokenized, and predicts the probabilites of every word in the vocabulary being the next token. We pass the essay through the LLM by growing the input sequence one character at a time, predicting the next characters probabilites, and comparing those to the actual next token in the essay. We can then see the ordinality of the actual token as predicted by an

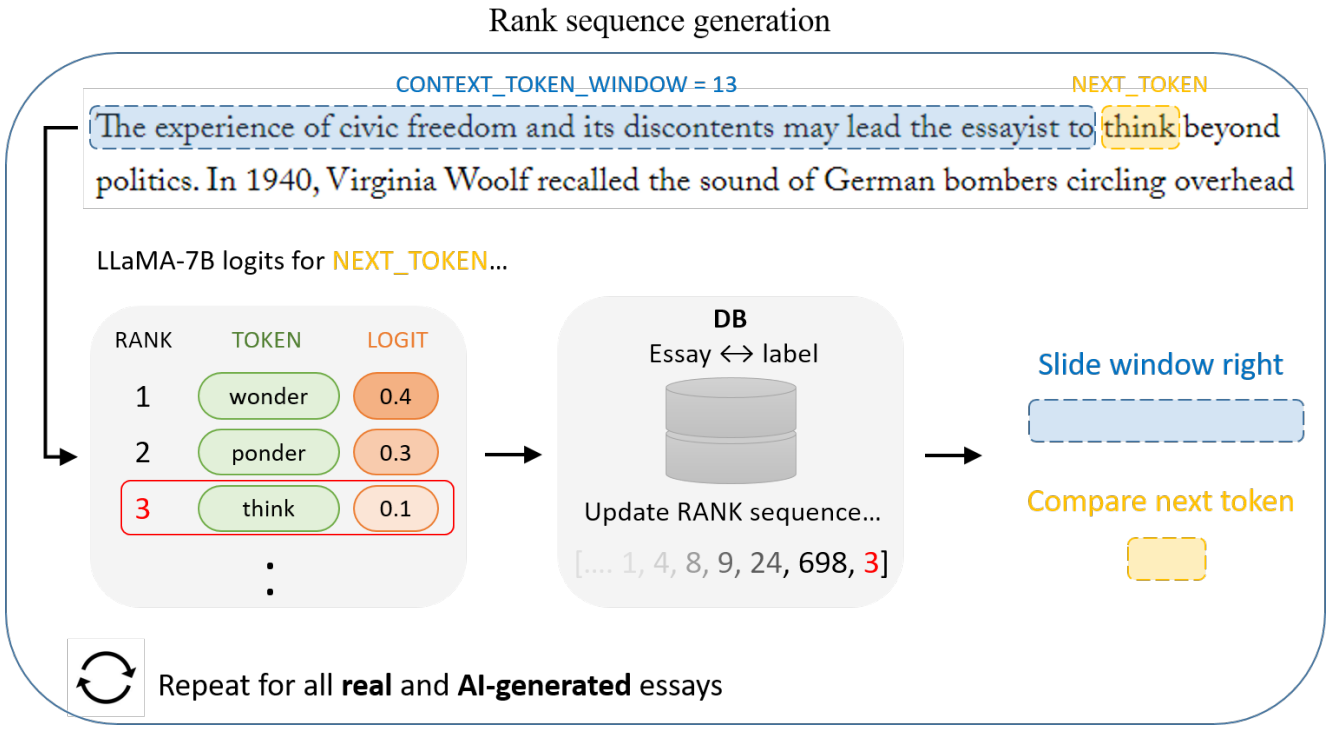


Figure 2. Rank sequence generation using LLaMA-7B involves extracting a window of tokens from input document which is provided as input to LLaMA-7B to obtain the probable next token logit distribution. These rank sequences are stored according to their label for generating $P(D_{real})$ and $P(D_{fake})$ later.

LLM! If you put the tokens in order of descending probability, you can see how likely the actual next token is, given the preceding tokens. If the next token is the highest ranked token per the LLM output, it is exactly what the LLM predicted. As the ranking increases, the less likely the word is and is more "surprising". Our theory is that those surprises will be more less uniform in individual generated essays, compared to a temperature dependent pseudo-surprise LLM generated essay.

### 1.3. Modelling

The task is a binary classification, given an essay we predict whether it was AI generated, or individual generated. We plan to pursue two avenues. We plan to develop an unlearned method that uses KL divergence. We believe that the differing distribution of the ordinality of chosen words expressed as a histogram will look different for the AI vs. individual generated essays. Our learned method will take these histograms as inputs and classify them. Llama has a vocabulary size of 32,000. We will be using a cutoff as a hyperparameter, putting all of the selected words greater than this number into one final bin.

## 2. Results

In this section, we present the results from two approaches: i) our proposed method, and ii) an alternative approach that only utilizes an NN working on text embeddings. This was done as we wanted an alternative approach baseline.

### 2.1. Our approach

While Fig. 2 mentions using both, a KLD module and a neural network, during our experiments, we noticed that the KLD method did not perform too well. Hence, we omit result analysis for KLD. We experimented with several preprocessing methods. The first was the simplest, we took the binned llama outputs and simply normalized them for the length. With 200 bins and 200 epochs we were able to get to 92 percent train accuracy. We tried taking the differences between each bin and the previous bin; The KLD results made us think that a less smooth distribution would be more indicative of a human generated essay and we thought this would better capture that. With 200 epochs and 200 bins we achieved 93.26 percent training accuracy. Finally we used products. We took each bin and multiplied it by all bins greater than it (resulting in no duplicate products). The interaction sequence $\mathbb{I}$ that we obtain from the histogram, or the normalized frequency vector $\mathbf{v}$ can be seen in Algorithm 2. Our resultant interaction sequences had 19,900

individual products. This training accuracy rose to 99.76 percent train accuracy, and our validation accuracy got to 92.55 percent (early in our 200 epochs). The architecture of the fully-connected network we use is given in Table 1. Additionally, we make use of the Adam optimizer with a learning rate of 1e-3 and a batch size of 32.

**Algorithm 2** Interaction sequence generation

**Input:** normalized frequency vector $\mathbf{v}$
**Output:** interactions vector $\mathbb{I}$

$\mathbb{I} \leftarrow \{\}$
**for** $i \in 1 \rightarrow |\mathbf{v}|$ **do**
  **for** $j \in i \rightarrow |\mathbf{v}|$ **do**
    $\mathbb{I} \xleftarrow{+} v[i] \times v[j]$
  **end for**
**end for**

Table 1. Fully-connected network architecture

| # | Layer |
|---|---|
| 1 | Linear, 200 |
| 2 | Linear, 100 |
| 3 | Linear, 1 |

We present the training history of our approach in Fig. 3. It is evident that the model starts to overfit after about 35 epochs as the validation loss starts to increase. While the model reaches a perfect training accuracy by the end of 200 epochs, the final validation accuracy hovers around 91%.

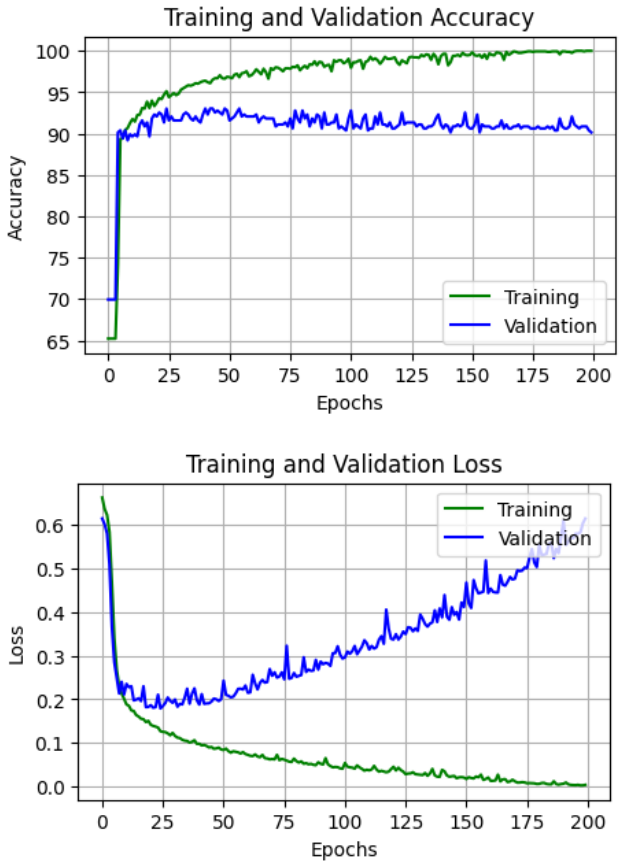


Figure 3. Training history of proposed approach.

## 2.2. Alternative approach

In Method 2.1, we explore an advanced classification strategy for distinguishing between human-written and AI-generated essays. This method relies on analyzing the probability distribution of subsequent word predictions, offering a nuanced approach to textual analysis. However, the complexity and length of this process render it impractical for evaluation on Kaggle's concealed dataset.

To address these challenges, we pivot to an alternative methodology involving a neural network and feature engineering. This network is adept at constructing class-specific probability distributions, streamlining the classification process. Our approach begins with meticulous feature engineering. We first cleanse the text data by eliminating newline characters, thus enhancing the dataset's integrity. Subsequently, we employ the TfidfVectorizer for text vectorization. This tool, configured to tokenize text into unigrams and bigrams and to strip accents, is uniquely fitted on the test data. This unsupervised approach circumvents the need for labels, allowing us to apply the vectorizer to the training data. The result is a conversion of text into a numerical format, ideally suited for machine learning applications. The prepared data offers a detailed feature vector for each essay, which we then introduce to our neural network. The network comprises three linear layers and utilizes the ReLU activation function. This structure forms the backbone of our classification model, tasked with differentiating between human and AI essay compositions. In testing, this model achieved a remarkable validation accuracy of 98.75%. However, when applied to Kaggle's hidden dataset, the accuracy dipped to 71.6%. This discrepancy is justifiable, given the dataset's divergence from our training data. While our training data encompassed essays from only two out of seven prompts, the test data spanned all seven, presenting a significantly broader domain.

To enhance accuracy, we propose the integration of more complex and diverse features, such as higher-order N-grams (e.g., trigrams, quadrigrams). However, we must balance this with computational feasibility, particularly in the context of this course project. Despite these limitations, our method demonstrates a sophisticated and methodical approach to neural network-based text analysis and classification.

## 3. Conclusion and Future Directions

This study presents novel solutions in the realm of AI-generated text detection. The proposed method is adversarial in nature, in that it utilizes an adversarial LLM to generate a logit distribution for an input sequence. The logit distributions of an LLM is the average results of all its training data, and logically generates smoother distributions. We saw these consistently and were able to capitalize on the fact to generate high classification accuracies. However, overfitting remains an issue. Even with the data that we sourced, our validation accuracy was unable to rise to an overtly reliable level.

Due to the limited domain of the competition dataset, we were not suprised at the overfitting. We believe our methodology would benefit from additional data. There is a convenient benefit from the nature of the problem; generating additional AI generated essays is fairly easy. Adding user generated essays does pose more of a challenge, but fine-tuning on specific datasets is feasible and could result in a useful specialized AI detection tool.

Our experiments with the proposed approach showed that our method is not scalable to mobile devices. Generating rank sequences from an LLM for each essay took 20-30s on a Tesla V100-SXM2-16GB GPU, and doing so on devices with fewer resources may take much longer. Future work includes making this approach more scalable.

As for intelligibility, our method achieves a highly explainable model. With rank sequences output by the LLM, we have **explainability at the word-level**, which enables practitioners to know how likely each input token was to have been generated by an LLM.

## 4. Contributions

- Dallin set up Llama and downloaded the weights onto the SCC. It requires an 8GB GPU to load the model, tokenize and run inference. The 1300 individual made essays as well as our supplemental AI generated essays have been passed through our ordinality process. These are stored in csv. Dallin wrote the training and validation code for the Llama histogram output classifier. This includes the code for the histogram binning, differences, and interactions we examined. We collaboratively tuned the hyper parameters for this model.
- GM played a significant role in refining the rank sequence generation concept in collaboration with Dallin's initial proposal of the entire idea. His notable contributions include the development of a production-grade KLD distance-based distribution discriminator, which is publicly accessible on our GitHub repository in the form of a meticulously structured and rigorously unit-tested package named 'KLDivergence'. In addition to his technical contributions, GM undertook the responsibility of generating all the high-quality figures presented in this document. Furthermore, he authored the mathematical framework that underpins the entire methodology. GM's multifaceted involvement has been instrumental in the success of this project so far.
- Himanshu's pivotal role in our project encompasses a wide array of vital contributions that significantly elevate our work. He is responsible for data generation from Large Language Models (LLM), employing strategies to effectively reduce bias in our training data. His expertise extends to data preprocessing, where he adeptly utilizes techniques such as regex and tokenization to enhance the quality of the generated text. Recognizing the importance of benchmarking, Himanshu has initiated and manages our Kaggle team for relevant competitions, fostering an environment of competitive learning and progress. Furthermore, he has established and meticulously maintains our project's GitHub repository, ensuring organized code management and seamless collaboration among developers. Adding to these significant roles, Himanshu has also developed the neural network architecture and feature engineering pipeline for this project. His work includes training a classification model on the provided training data and rigorously testing this model by submitting it to Kaggle competitions. He undertook the comprehensive process of understanding the submission format and implemented necessary changes for successful submission on Kaggle, demonstrating a thorough understanding of the competitive landscape and the technical requirements for successful project implementation.

## References


[1] GM Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020. 2

[2] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, pages IV–317. IEEE, 2007.

[3] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *CoRR*, abs/1404.2000, 2014. 2

[4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. 2