Detecting AI-Generated Essays

Annita Liu, Beichen Zhai, Hongkai Zhang, Xinmei Feng

Abstract

With the rise of large language models (LLMs), distinguishing human-written from AI-generated essays has become crucial. This study evaluates three approaches: statistics-based methods, traditional machine learning, and transformer-based deep learning, among which BERT demonstrated the best performance. To enhance utility, we explored accuracy improvements, distillation, and parameter-efficient fine-tuning (PEFT). Distillation reduced computational time with minimal performance loss, while PEFT achieved efficiency with a slight trade-off in performance. Error analysis revealed key precision-recall trade-offs, emphasizing the challenges of this task. This work highlights the importance of combining traditional and modern Natural Language Processing (NLP) techniques to enhance detection efficiency and robustness.

1 Introduction

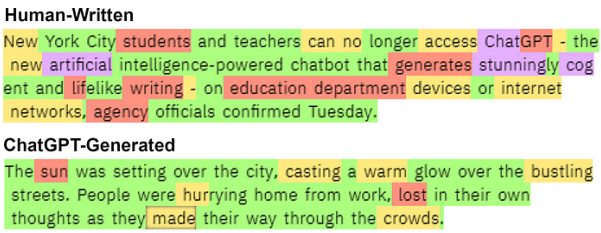

Figure 1: Token Predictability Rankings in Texts (Greener indicates higher predictability)

With the rapid advancements in LLMs such as GPT and Llama, distinguishing between human-written and AI-generated text has become a significant challenge in NLP. AI produces fluent, coherent, and contextually appropriate text that often closely mimics human writing, making it essential to identify the subtle linguistic and structural differences that set them apart.

The task in our term project focuses on developing classifiers to distinguish between human-written and AI-generated essays by leveraging three approaches: statistics-based method, traditional machine learning method, and transformer-based deep learning method. Using these methods, the task explores essential computational linguistic techniques for analyzing and modeling natural language. It also underscores the interplay between statistical tools, traditional machine learning techniques, and modern transformers, offering valuable insights into the rapidly evolving landscape of NLP methodologies and applications.

To illustrate, Figure 1 visualizes the predictability differences between AI-generated and human-written text. The ChatGPT-generated text contains a higher proportion of predictable tokens, indicated by green and yellow highlights, whereas the human-written text exhibits greater variability, with more unpredictable tokens in red and purple. These structural distinctions enable the differentiation between the two, underscoring the feasibility of our task.

Formally, given a dataset of essays, the task is to develop a classifier $f(x)$ that maps an input essay x to a binary label y, where $y = 0$ indicates a student-written essay and $y = 1$ represents an AI-generated essay. The classifier's performance will be evaluated using accuracy and F1 score, along with insights provided by error analysis and confusion matrices.

This task was selected for two primary reasons. First, it addresses real-world concerns about the influence of LLMs on education, with significant implications for maintaining academic integrity and preventing the misuse of generative AI. Second, this task incorporates various NLP methodologies, embodying a multidisciplinary approach that provides a comprehensive understanding of the evolving landscape of NLP techniques. The multidisciplinary nature of this task allows us to holistically apply the knowledge gained throughout the course to practical, real-world challenges.

2 Literature Review

The detection of AI-generated text has become a critical area of research in NLP. Several studies have proposed diverse methodologies to address this challenge, each contributing unique advancements.

Wu et al. (2023) provided a comprehensive survey examining various techniques for detecting AI-generated text, categorizing detection methods into several approaches: synonym substitution-based methods, sequence-to-sequence methods, statistics-based methods, neural-based classifiers, black-box and perturbation-based methods, and human-assisted and mixed methods.

Building on this foundational understanding, specific works focused on detailed techniques for text detection. For example, Gehrmann et al. (2019) introduced GLTR (Giant Language Model Test Room), a statistics-based method for detecting LLM-generated text. GLTR analyzes the likelihood of each token being generated by a language model (e.g., GPT-2) using average token probability. It detects artifacts in text generation by examining whether tokens are sampled from the top of the model's probability distribution. Additional features, such as token ranking and entropy, further enhance detection capabilities. GLTR's effectiveness was demonstrated in a human-subjects study: without the GLTR interface, participants achieved 54.2% detection accuracy, only slightly better than random guessing. However, with GLTR visualizations, accuracy increased to 72.3%, showcasing its utility in improving human judgment.

In addition to statistics-based methods, deep learning is a widely used approach for this task. Wang et al. (2024) introduced a detection model based on the BERT algorithm, utilizing a broadly balanced private dataset consisting of 708 AI-generated texts and 670 human-authored texts. Before being fed into the BERT model, the dataset underwent extensive preprocessing, which included converting text to lowercase, splitting words, removing stop words, applying stemming, removing digits, and eliminating redundant spaces to improve data quality and accuracy. The model demonstrated strong generalization capabilities, achieving an average accuracy of 97.71% on the test set.

Expanding on BERT-based research, Abassy et al. (2024) highlighted the utility of DistilBERT for detecting machine-generated texts. Compared to other detectors like RoBERTa, DistilBERT offers significant advantages in speed and size—being 60% faster and 40% smaller while retaining approximately 97% of BERT's language understanding capabilities. Moreover, the results suggest that DistilBERT may surpass RoBERTa on specific datasets. For instance, in the OUTFOX dataset, DistilBERT achieved a competitive performance with a 96.65% F1 score, outperforming RoBERTa's 95.53%. This underscores DistilBERT's potential as a lightweight yet effective alternative in domain-specific applications.

Beyond distillation, an increasing number of studies have explored fine-tuning techniques to enhance model adaptability and minimize computational resource requirements. Tolstykh et al. (2024) proposed GigaCheck, a framework for AI-text detection that utilized a fine-tuned Mistral-7B model enhanced with LoRA to improve parameter efficiency. GigaCheck demonstrated high performance across various datasets, including TweepFake and TuringBench, outperforming existing models like GLTR, BERT, and RoBERTa. Notably, it achieved a 94.3% F1 score on TweepFake and 99.66% F1 on TuringBench. Additionally, GigaCheck applied embeddings from Mistral-7B with the DN-DAB-DETR detection architecture to identify AI-generated text segments within collaborative human-machine documents, achieving up to 13% higher accuracy than BERT-based models.

3 Experimental Design

3.1 Data

This section describes the training, development, and test data used in this project.

The dataset used in this project is sourced from Kaggle (Thite, 2024). The dataset was created to support research in distinguishing between human-written and AI-generated text, and detailed information about its construction is available in the original Kaggle repository.

Table 1: Dataset Composition

	Count	**Proportion (%)**
Human-written (Label = 0)	17,508	60%
AI-generated (Label = 1)	11,637	40%
Train Set	23,316	80%
Development Set	2,914	10%
Test Set	2,915	10%
Total	**29,145**	**100%**

The dataset comprises a total of 29,145 essays. Of these, approximately 60% are authored by stu-

dents (17,508 entries), while the remaining 40% are generated by LLMs (11,637 entries).

The dataset is partitioned into three subsets: the training set contains 23,316 essays (80%), the development set includes 2,914 essays (10%), and the test set consists of 2,915 essays (10%). Each entry in the dataset includes the essay text and a corresponding target label: 0 for human-written and 1 for AI-generated. The data is stored in a structured CSV format to facilitate efficient loading and processing. An example from the dataset is shown in Table 2.

It is worth noting that the dataset exhibits a slight skew in label distribution, with a higher proportion of student-authored essays (60%) compared to those generated by LLMs (40%). This imbalance may introduce subtle, although not many, biases during model training.

Table 2: Example Entries

Essay Text	Label
The art of technology to read expression would be valuable for many reasons. Many people like to guss or are unaware of peoles feelings which could cause problem with communication. It alsp could stop some problemd not all but some...	0
In my opinion, the work world has become more stressful and less leisurely than it used to be. In the past, people could take their time and relax at home. Now, people are often working all day and then having to go back to work the next day. This can be very demanding and stressful. People are also working in different locations and at different times of the day. This can be very difficult to get a good night's sleep...	1

3.2 Evaluation Metric

Given that the output is binary, we employ the following metrics to evaluate the performance of our models: accuracy, precision, recall, and F1.

Accuracy measures the overall correctness of the model and is calculated as the ratio of correctly classified essays to the total number of essays. Mathematically, accuracy is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

Precision quantifies the proportion of correctly identified positive samples out of all samples predicted as positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall, also known as sensitivity, measures the ability of the model to identify all actual positive samples:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The F1 score provides a harmonic mean of precision and recall, balancing the trade-off between the two:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics have been extensively used in tasks like text classification and detection, as shown in prior language modeling studies (Wang et al., 2019; Zellers et al., 2019). By using these standard evaluation criteria, we ensure comparability with past research and a robust analysis of our model's capabilities.

3.3 Simple Baseline

This section evaluates the task's difficulty by reporting the majority class baseline, which predicts the most frequent class from the training data for all test samples.

Steps: 1. Determine the majority class from the training data; 2. Predict the majority class for all test samples.

Evaluation Results: The performance of the majority class baseline is shown in Table 3. The confusion matrix in Table 4 demonstrates the model's tendency to predict only the majority class.

Although it does not provide a non-trivial benchmark for precision, recall, or F1 score due to its nature, the evaluation metric results demonstrate that the simplest and most naive approach achieves an accuracy of ~61%, setting a moderate benchmark for this task—not too low, but not overly high.

Table 3: Evaluation Results for Simple Baseline

Metric	Value
Accuracy	60.79%
Precision	0.00%
Recall	0.00%
F1 Score	0.00%

Table 4: Confusion Matrix for Simple Baseline

	Predicted 0	Predicted 1
Actual 0	1772	0
Actual 1	1143	0

Table 5: Evaluation Results for GLTR

Metric	Value
Accuracy	62.02%
Precision	80.00%
Recall	4.2%
F1 Score	7.98%

Table 6: Confusion Matrix for GLTR

	Predicted 0	Predicted 1
Actual 0	1760	12
Actual 1	1095	48

4 Experimental Results

4.1 Stronger Baselines

4.1.1 Statistical Detection

Our initial published baseline adopts a GLTR-like methodology from Gehrmann et al. (2019), leveraging a statistics-based approach. This method uses token probabilities from the GPT-2 language model to identify patterns indicative of AI-generated text. Specifically, it evaluates whether text tokens exceed a predefined probability threshold to classify the text as machine-generated.

Steps:

1. Tokenize the text using GPT-2's tokenizer.

2. Calculate the probability of each token under GPT-2 using the formula:

$$P(\text{token}_i) = \frac{\exp(\text{logit}_i)}{\sum_j \exp(\text{logit}_j)}$$

If the probability of a token exceeds 0.5, we classify it as having a high probability of being generated by a machine.

3. Tune the threshold for the proportion of tokens with high probability to determine whether an essay is AI-generated. This hyperparameter was optimized on the development dataset within the range of 20% to 70%, with a step size of 5%. The best threshold was found to be 40% (based on F1).

4. Predict the labels for test samples based on the proportion of tokens exceeding the threshold:

$$\text{Prediction} = \begin{cases} 1, & \text{if } \frac{\sum_{i=1}^{n} I(P(\text{token}_i) > 0.5)}{n} > 0.4, \\ 0, & \text{otherwise.} \end{cases}$$

Evaluation Results: The GLTR-like baseline achieves the performance metrics shown in Table 5. The confusion matrix in Table 6 highlights the model's improved accuracy, precision, recall, and F1 compared to the majority class baseline.

The results in this study differ from those reported in Gehrmann et al. (2019), where an accuracy of 72.3% was achieved—approximately 10% higher than the accuracy of our GLTR-like baseline (62.02%). This discrepancy arises because their approach utilized GLTR as an assistive tool to enhance human judgment, rather than as an automated classification system. With human intuition supported by GLTR visualizations, their method benefited from a nuanced assessment. In contrast, our approach fully automates the GLTR-like technique by applying a strict probability threshold for classification into labels 1 or 0. Considering this difference, a 10% gap is not overly significant.

That said, the relatively limited improvement of the GLTR approach over the benchmark, both in our study and in Gehrmann et al. (2019), underscores the constraints of relying solely on statistical properties for detecting AI-generated text. This highlights the necessity of adopting more advanced machine learning and deep learning techniques to capture the complex linguistic patterns and contextual nuances required for reliable detection.

4.1.2 Traditional Machine Learning

While the majority of research focuses on statistics-based and deep learning methods, our second baseline bridges the gap between the two by employing a traditional machine learning model. We employ logistic regression, as described in Kleinbaum and Klein (2010), as the primary classification model, relying on a diverse set of linguistic features to distinguish between AI-generated and human-written essays. The features include readability metrics, syntactic patterns, lexical richness, and spelling errors, all of which are designed to capture distinctive

patterns in text.

To derive these features, we utilized libraries such as `spacy`, `nltk`, and `textstat`. Readability metrics, including Flesch Reading Ease and Flesch-Kincaid Grade Level, provide insights into the complexity of the text. Syntactic features, such as the normalized distribution of part-of-speech (POS) tags, quantify the structural elements of language. Lexical richness is assessed using the Type-Token Ratio (TTR), which measures the variety of unique words in the text relative to its length. Additional features include the proportion of misspelled words, frequency of discourse markers like "however" and "moreover," as well as average sentence and word lengths. All features are normalized to account for essays of varying lengths, ensuring consistency in their interpretability.

The logistic regression model was trained using `scikit-learn` with standardized features to ensure balanced contributions across all inputs. To prevent overfitting, we incorporated a validation split during training to tune hyperparameters and evaluate generalization performance. The model converged within 1000 iterations, which confirms the stability of the solution.

Evaluation Results: The performance metrics for logistics regression classifier baseline are shown in Table 7 with the confusion matrix in Table 8.

Table 7: Evaluation Results for Logistics Regression Classifier

Metric	Value
Accuracy	93.76%
Precision	92.64%
Recall	91.34%
F1 Score	91.98%

Table 8: Confusion Matrix for Logistics Regression Classifier

	Predicted 0	Predicted 1
Actual 0	1689	83
Actual 1	99	1044

The logistic regression classifier significantly outperforms both the simple majority baseline (60.79% accuracy) and the GLTR method (62.02% accuracy, 7.98% F1 score), achieving 93.76% accuracy and an F1 score of 91.98%. By leveraging engineered features, it effectively distinguishes between human-written and AI-generated text, with high precision (92.64%) and recall (91.34%) for detecting AI content, as shown in its balanced confusion matrix.

However, the reliance on manually engineered features remains a limitation of this approach. While these features capture specific linguistic patterns, they may still struggle to represent the nuanced complexities of language inherent in more diverse or sophisticated datasets. Moreover, the model's strong performance is closely tied to the distribution of the training dataset, raising concerns about its ability to generalize effectively to unseen data with differing characteristics. Despite these constraints, the logistic regression classifier establishes a robust and meaningful baseline, serving as a foundational approach before delving into more advanced methods, such as transformers, in the next section.

4.1.3 Transformer

Our final baseline is a published approach that builds on the work of Koroteev (2021) and Wang et al. (2024). We employed a BERT model fine-tuned for binary classification from HuggingFace's Transformers library. We performed full fine-tuning of this BERT model on our downstream dataset with a learning rate of 5×10^{-5}, a batch size of 8, and for 3 epochs. The AdamW optimizer was used during training, with the evaluation strategy configured to assess the model at the end of each epoch.

The fine-tuning pipeline utilized the HuggingFace Trainer API, which streamlined model training and evaluation. The training and validation datasets were tokenized using BERT's tokenizer with a maximum sequence length of 512 tokens. Padding and truncation ensured consistency across inputs.

The evaluation results on the test set in Table 9 and 10 reveal a substantial improvement over GLTR, with BERT outperforming logistic regression by an 8% higher F1 score.

Table 9: Evaluation Results for BERT

Metric	Value
Accuracy	99.86%
Precision	99.65%
Recall	100.00%
F1 Score	99.83%

Our high performance exceeds those reported by Wang et al. (2024) (97.71%) by approximately

Table 10: Confusion Matrix for BERT

	Predicted 0	**Predicted 1**
Actual 0	1768	4
Actual 1	0	1143

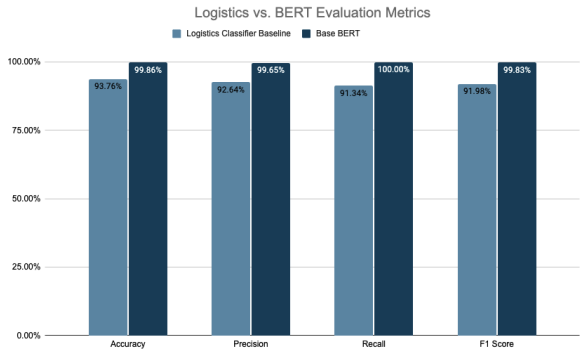

Figure 2: Logistics Baseline vs. BERT Baseline

2%. This improvement is likely attributed to two key factors. First, Wang et al. (2024) employed extensive text preprocessing, such as removing stop words, punctuation, unnecessary spacing, and other elements, before inputting the essays into the BERT model. We believe that retaining unprocessed texts helps capture subtle differences between human and AI-generated content. For instance, unprocessed texts may retain distinctive features such as typos, irregular spacing, or the abrupt use of stop words in human-written texts, which their preprocessing may have removed. By using raw essays as input, our approach preserves these potentially informative signals. Second, their use of a different dataset may introduce variations in data quality compared to ours, potentially contributing to the observed performance differences.

That said, the strong results of our BERT baseline (100% recall) raised concerns about whether the outcomes are due to a lack of diversity and real-world complexity in the training dataset we used. To address this, we incorporated a robustness check in Appendix A, evaluating the trained model on a complex and diverse external dataset. This introduced greater variability and tested its robustness in diverse scenarios. The results demonstrate that the model remains robust when evaluated on a more complex external dataset.

4.2 Extensions

We performed three extensions on logistic regression and BERT but did not extend GLTR, as we believed statistical methods alone could not achieve high evaluation scores.

Our logistic regression extension aimed to improve accuracy and F1, but we saw limited benefit in applying similar enhancements to BERT, which already performed exceptionally well. However, full fine-tuning of BERT for 3 epochs took 7 hours on T4 GPU, making it impractical for low-resource systems. To address this, we implemented Distillation and PEFT, significantly reducing training time while maintaining comparable evaluation scores to the baseline.

4.2.1 Augmentation, Features, and Ensembles

We identified potentials for improving logistic regression performance, considering its sensitivity to feature representation, parameter optimization, and susceptibility to overfitting. To address this, we implemented data augmentation, expanded linguistic feature sets, conducted hyperparameter tuning, and incorporated an ensemble method with a Random Forest classifier. For data augmentation, targeted techniques like synonym substitution (WordNet) and character perturbations (random insertions, deletions, and swaps) were applied to 20% of the training data to increase robustness. For feature extraction, we incorporated diverse linguistic features, including readability metrics (Flesch Reading Ease, **Coleman-Liau Index**), spelling error rates, POS distributions, discourse marker frequencies, TTR, sentence/word lengths, and **GPT-2 perplexity scores**. For hyperparameter tuning, using grid search with 3-fold cross-validation, we optimized logistic regression, selecting a regularization strength $C = 1$ and the L2 penalty. Finally, to further improve classification accuracy, we combined the tuned logistic regression model with a Random Forest classifier, based on Rigatti (2017). Probabilities from both models were averaged with a threshold of 0.5 for final predictions, leveraging the precision of logistic regression and the robustness of Random Forest.

Table 11: Performance Comparison between Baseline and Enhanced Logit

	Accuracy	**Precision**	**Recall**	**F1**
Baseline	93.76%	92.64%	91.34%	91.98%
Enhanced	98.94%	98.52%	98.78%	98.65%

As shown in Table 11, the ensemble model significantly outperforms the logistic regression baseline, achieving an accuracy of 98.94% and an F1 score of 98.65%. This represents a notable im-

Table 12: Confusion Matrix for Enhanced Logit

	Predicted 0	**Predicted 1**
Actual 0	1755	17
Actual 1	14	1129

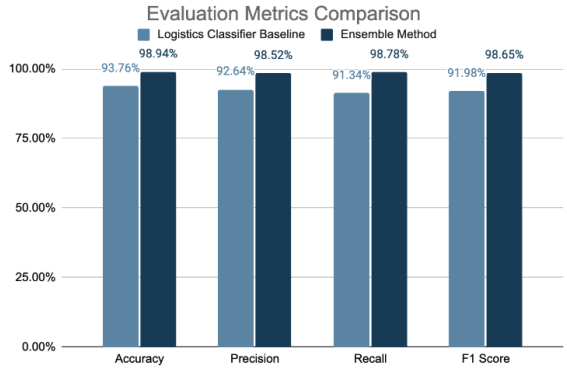

Figure 3: Logistics Baseline vs. Ensemble Method

provement over the baseline's 93.76% accuracy and 91.98% F1 score, with a 5.18% increase in accuracy and a 6.67% increase in F1 score.

By integrating hyperparameter-tuned logistic regression with a Random Forest classifier and leveraging additional features, the ensemble effectively reduces misclassifications from 182 to just 31, as shown in Table 12. This demonstrates the power of combining models to capture complex patterns that a single model may miss, and the ensemble approach underscores the value of feature expansion and model diversity in achieving near-optimal detection performance.

4.2.2 Distillation

Although the transformer baseline's results already achieved state-of-the-art performance, it highlights a key limitation: the high computational cost and time required to fully fine-tune large transformer models. To address this limitation, we implemented a distillation-based extension builds upon the work of Abassy et al. (2024), which demonstrated that DistilBERT can serve as a powerful yet efficient tool for detecting AI-generated text. Our approach uses knowledge distillation to transfer the performance of a BERT model (teacher) to a smaller and faster DistilBERT model (student), significantly reducing training and inference time while maintaining competitive performance.

In this extension, we utilized a pre-trained BERT model fine-tuned for classification as the teacher and a pretrained DistilBERT model as the student. The student model was trained using a combination of the following:

1. **Hard Labels**: Cross-entropy loss computed with ground truth labels.

2. **Soft Labels**: KL divergence loss computed with the teacher model's probability distribution over classes, scaled by a temperature factor of 2.0.

The distillation loss combines these two components with a weighting factor ($\alpha = 0.5$) to balance hard and soft supervision:

$$L_{\text{distillation}} = \alpha \cdot L_{\text{KL}} + (1 - \alpha) \cdot L_{\text{CE}}$$

The training setup was optimized for efficiency while maintaining high performance, using the AdamW optimizer with a learning rate of 5×10^{-5}, a batch size of 8, and running for 3 epochs. During training, the teacher model was kept frozen to reduce computational overhead, while the student model's parameters were updated to mimic the teacher's outputs.

The evaluation metrics in Table 13 reflect the high performance of the distilled model. Table 14 further illustrates the model's balanced performance across both classes, with only 12 misclassifications in a test set of 2,915 samples.

Table 13: Performance Comparison between BERT and DistilBERT

	Accuracy	**Precision**	**Recall**	**F1**
BERT	99.86%	99.65%	100.00%	99.83%
DistilBERT	99.59%	99.82%	99.13%	99.47%

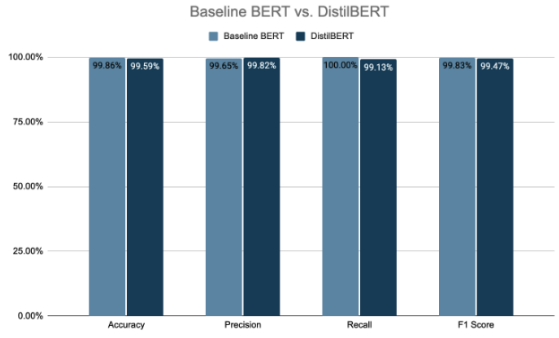

Figure 4: Baseline BERT vs. DistilBERT

The distilled model represents a notable improvement in utility over the baseline BERT model, offering similar evaluation scores with substantially reduced computational requirements. The baseline BERT model achieved outstanding performance,

Table 14: Confusion Matrix for DistilBERT

	Predicted 0	**Predicted 1**
Actual 0	1770	2
Actual 1	10	1133

but it required approximately 7 hours to fine-tune on a T4 GPU. In contrast, the distilled model delivered competitive performance, achieving an accuracy of 99.59% and an F1 score of 99.47%, closely matching BERT's results in Table 13, while completing training in under 4 hours. By reducing training time by 43% and maintaining near-state-of-the-art performance, DistilBERT emerges as an efficient and effective choice for low-resource systems.

4.2.3 PEFT

In addition to distillation, which optimizes the model itself, we explored PEFT methods that focus on improving efficiency by operating on parameter updates, inspired by Tolstykh et al. (2024). We implemented LoRA (Hu et al., 2021), which introduces trainable low-rank matrices into each layer of the Transformer architecture, significantly reducing the number of trainable parameters. We also applied its variant, Weight-Decomposed Low-Rank Adaptation (DoRA) (Liu et al., 2024), which decomposes pre-trained weights into magnitude and direction components, adapting them separately to enable fine-grained adjustments[1]. A detailed yet intuitive comparison between LoRA and DoRA architectures is provided in Appendix B.

To optimize computational resource efficiency, we applied PEFT to our DistilBERT model using the same training settings as the baseline BERT and DistilBERT models. For PEFT, we set the scaling factor to 32 and the rank to 8, minimizing the rank and, consequently, the size of the LoRA matrix. The F1 score on the test set and the reduction in GPU training load are reported in Table 15. Additional metrics, such as accuracy and the number of trainable parameters, are detailed in Table 16.

Table 15: Performance for PEFT

	F1	**%Params**	**Train Time**
BERT	99.83%	100.00%	7 hours
DistilBERT	99.47%	61.16%	$\leq$4 hours
LoRA	96.82%	0.68%	45 minutes
DoRA	97.03%	0.68%	53 minutes

[1]Quantization was not implemented in this work because quantization for DoRA has not yet been developed.

Table 16: Additional Performance Metrics for PEFT

	Accuracy	**Precision**	**Recall**	**#Params**
BERT	99.86%	99.65%	100.00%	109,483,778
DistilBERT	99.59%	99.82%	99.13%	66,955,010
LoRA	97.43%	93.91%	99.91%	739,586
DoRA	97.60%	94.30 %	99.91%	748,802

Table 15 highlights the remarkable efficiency of PEFT methods, with both LoRA and DoRA requiring only 0.68% of the trainable parameters of the baseline model, reducing training loads by over 100 times. Additionally, GPU training time was reduced to less than one-seventh of the baseline, all while sacrificing only about 2-3% of F1 performance compared to baseline BERT. This minimal trade-off demonstrates the exceptional efficacy of PEFT as a tool for efficient language modeling when paired with well-tuned hyperparameters. Notably, while DoRA requires nearly the same computational resources as LoRA, it achieves a higher F1 score, underscoring the benefits of finer adjustments and decomposition techniques for maximizing performance with resource constraints.

Interestingly, Table 16 shows that the loss of performance compared to full fine-tuning is primarily due to a drop in precision. This may result from the reduced model capacity, which prioritizes the primary goal of detecting AI-generated text (high recall) while compromising its ability to accurately distinguish human-written text.

4.3 Error analysis

The BERT family models, including the baseline BERT and DistilBERT, exhibit state-of-the-art performance, yet they make distinct types of errors that reveal inherent trade-offs in their design and training. This section categorizes the errors made by both models, evaluates their prevalence, and compares their strengths and weaknesses.

4.3.1 Baseline BERT Model

The baseline BERT model exhibited exceptional performance, achieving an accuracy of 99.86% and an F1 score of 99.83%, with only 4 misclassified samples out of 2,915 (0.14%). The following examines the specific patterns in the errors it made:

1. **Error Types:**
    - False Positives (*Human-written* $\rightarrow$ *AI-generated*): All misclassifications in the baseline BERT model were false positives, where human-written text was incorrectly labeled as AI-generated. This

indicates that the model struggled to distinguish between authentic human-like writing and text exhibiting patterns that resemble machine-generated content.
- Prevalence: 100% of the errors were false positives.

2. **Examples of Misclassified Samples:**
   - Sample 202 (False Positive):
     - Text: "In recent years, many places all around the world have taken the initiative to lower the use of motor vehicles, specifically cars, with the intent of decreasing the amount of greenhouse gas being emitted..."
     - Actual Label: Human-written (0)
     - Predicted Label: AI-generated (1)
     - Analysis: The structured argument and formal tone of this text closely mimic AI-generated patterns, leading to misclassification.
   - Sample 771 (False Positive):
     - Text: "Since the early societies with important philosophers such as Aristotle and Confucius, the world has been in a complex debate arguing about the best way to influence other people..."
     - Actual Label: Human-written (0)
     - Predicted Label: AI-generated (1)
     - Analysis: The philosophical discussion and use of formal language contributed to the model's incorrect prediction.

The errors suggest that the baseline BERT model's high recall (100%) for detecting AI-generated content comes at the cost of reduced precision when human-authored texts exhibit machine-like characteristics. These errors are relatively rare, but they highlight the challenge of identifying nuanced distinctions between complex human-written and AI-generated text, especially when human-authored content exhibits an overly formal and objective tone, which can blur the line between the two. This also raises a question: can the model effectively detect tonal nuances when humans intentionally mimic LLMs, presenting an adversarial attack?

4.3.2 Distilled BERT Model

The DistilBERT model, while slightly less accurate than the baseline, achieved impressive results with an accuracy of 99.59% and an F1 score of 99.47%, misclassifying 12 samples out of 2,915 (0.41%). The errors made by the distilled model exhibited a broader variety compared to the baseline.

1. **Error Types:**
   - False Positives (*Human-written* $\rightarrow$ *AI-generated*): Prevalence - 17% (2 out of 12 errors). Similar to the baseline, the distilled model sometimes mislabeled human-written text as AI-generated, especially when the text exhibited formal tone and structured reasoning.
   - False Negatives (*AI-generated* $\rightarrow$ *Human-written*): Prevalence - 83% (10 out of 12 errors). The majority of errors were false negatives, where AI-generated text was misclassified as human-written. These errors suggest a slight drop in recall compared to the baseline model, likely due to the distilled model's reduced capacity.

2. **Examples of Misclassified Samples:**
   - Sample 299 (False Negative):
     - Text: "I think drivers should not be able to use cell phones while driving. It's really dangerous and a lot of people get injured or even killed from accidents caused by distracted driving..."
     - Actual Label: AI-generated (1)
     - Predicted Label: Human-written (0)
     - Analysis: The conversational and opinionated tone might have led the distilled model to perceive this as human-written text.
   - Sample 360 (False Negative):
     - Text: "Hey, I know that some people believe that the Face on Mars is proof that aliens visited our planet, but let's get real here. There's no way that some aliens carved a giant face into the side of a mountain..."
     - Actual Label: AI-generated (1)
     - Predicted Label: Human-written (0)
     - Analysis: The informal tone and skepticism expressed in this text likely contributed to the misclassification.

While the distilled model significantly reduced false positives, it introduced additional false negatives. This highlights a trade-off between precision and recall, where the distilled model slightly prioritized precision over recall.

4.3.3 Comparison Between Models

The distilled model successfully corrected some false positives made by the baseline BERT model, demonstrating improved generalization for human-written text with machine-like patterns. However, its reduced capacity led to an increase in false negatives, where AI-generated content with informal tones was misclassified as human-written.

In summary, DistilBERT's balance of performance and computational efficiency, at the expense of a slight emphasis on precision over recall, makes it a valuable choice for resource-constrained environments. On the other hand, the baseline BERT model remains superior in recall, making it more reliable for scenarios where detecting AI-generated content is paramount.

5 Conclusions

In this term project, we tackled the challenging task of distinguishing human-written text from AI-generated text by employing a statistics-based method (GLTR), traditional machine learning (logistic regression), and transformer-based deep learning (BERT). Among these approaches, BERT achieved near state-of-the-art results, with an accuracy of 99.86% and an F1 score of 99.83%. This highlights the power of modern transformers in detecting nuanced differences between human and machine-generated text.

To address the computational challenges of fine-tuning large models, we implemented DistilBERT, which reduced training time by 43% while maintaining strong performance, achieving an F1 score of 99.47%. Additionally, PEFT methods such as LoRA and DoRA reduced computational requirements by fine-tuning only 0.68% of BERT's parameters, with just a 2–3% trade-off in F1 performance. These techniques demonstrate the feasibility of employing transformer models in low-resource environments without significant performance loss.

We also explored feature augmentation and model optimization using ensemble techniques, which improved the F1 score of logistic regression by approximately 6%. This underscores the value of enhancing traditional methods to tackle challenging and nuanced tasks.

While BERT and DistilBERT delivered near state-of-the-art results, error analysis revealed challenges in balancing precision and recall, particularly for complex human-written texts mimicking machine-like patterns, and vice versa. Addressing these challenges necessitates further exploration of diverse datasets and linguistic styles to enhance generalization.

Future work could focus on improving model robustness through diverse datasets and adopting advanced techniques, such as quantization and adapters, to reduce computational costs. Integrating these advancements would enhance the practicality and effectiveness of AI-generated text detection systems, enabling broader real-world applications.

6 Acknowledgements

We sincerely thank our TA, Anqi Lin, for her invaluable guidance and support throughout this project. We extend our gratitude to Prof. Yatskar and the course, especially to the content on efficient language modeling, which provided foundational knowledge and inspired our work on distillation and PEFT extensions. We also acknowledge the auxiliary use of ChatGPT for tasks such as generating sample texts for augmentation, supporting brainstorming, and diagnosing error messages in the code, while ensuring its role remains minimal and strictly supplementary to our efforts. Additionally, we are grateful to the HuggingFace Transformers library for facilitating the implementation of our models, the creators of the Kaggle dataset for providing the foundation for this research, and the authors of referenced works for their inspiring contributions to the field.

7 Appendix

A Robustness Check

To assess the robustness of our model, we evaluated the performance of BERT on an external dataset comprising 6,026 essays, sourced from TheDRCat (2024). This dataset features a diverse range of topics and writing styles from both human and AI. The AI-generated texts were created using a variety of models, including Cohere Command, Google PaLM, and GPT-4, with enhanced prompts designed to increase complexity and diversity. These enhancements provided a rich and varied evaluation set, enabling us to rigorously test the model's ability to generalize to unseen, real-world data.

Table 17: Performance Comparison between Internal and External Test Sets

	Accuracy	Precision	Recall	F1
Internal	99.86%	99.65%	100.00%	99.83%
External	96.26%	91.32%	99.39%	95.19%

Table 18: Confusion Matrix for the External Test Set

	Predicted 0	Predicted 1
Actual 0	3563	234
Actual 1	15	2463

The results presented in Table 17 highlight the strong performance of our model, demonstrating its robustness even when applied to more complex data. The model achieved an accuracy of 96.26%, precision of 91.32%, recall of 99.39%, and an F1 score of 95.19%. Despite this impressive performance, as shown in Table 18, 249 misclassifications (4.13%) were observed. These errors stemmed from stylistic overlaps between human-written and AI-generated texts, highlighting the challenge of distinguishing nuanced patterns. This demonstrates the model's strong generalization to unseen data while identifying opportunities to improve handling of stylistically ambiguous cases.

B Comparison of LoRA and DoRA

Architecture

For simplification, we omit the scaling factor since it is identical in both LoRA and DoRA.

In LoRA, the updated weight matrix is computed as:

$$W_{\text{updated}} = W_{\text{pretrain}} + \delta W = W_{\text{pretrain}} + BA,$$

where B and A are low-rank matrices with dimensions defined by rank r. Specifically:

- B is initialized to zeros.
- A is initialized using the Kaiming distribution.

This formulation ensures that the additional parameters introduced by B and A are significantly smaller compared to the full weight matrix, making LoRA highly efficient.

In DoRA, the pre-trained weight matrix is decomposed into magnitude and direction components. The decomposition is expressed as:

$$W_{\text{pretrain}} = ||W_{\text{pretrain}}|| \cdot \frac{W_{\text{pretrain}}}{||W_{\text{pretrain}}||},$$

where:

- $||W_{\text{pretrain}}||$ is the magnitude vector m.
- $\frac{W_{\text{pretrain}}}{||W_{\text{pretrain}}||}$ is the directional matrix $\frac{V}{||V||}$.

To reduce training parameters, the norm of the directional matrix, $||V||$, is set to a constant C that is large enough to scale down the numerator during training. This simplification minimally affects performance.

The magnitude vector m is fully tuned since its size is relatively small. The low-rank adaptation in DoRA is applied only to the directional matrix, resulting in:

$$\frac{V}{||V||}_{\text{updated}} = \frac{W_{\text{pretrain}} + BA}{C},$$

where B and A are the same low-rank matrices as in LoRA. The updated weight matrix in DoRA is therefore:

$$W_{\text{updated}} = m_{\text{updated}} \cdot \frac{V}{||V||}_{\text{updated}}.$$

As demonstrated by Liu et al. (2024), DoRA allows for more fine-grained and nuanced parameter adjustments. It has proven to be highly robust, particularly with low-rank configurations (e.g., $r = 8$), and more effectively replicates the learning patterns of full fine-tuning compared to LoRA.

To clarify the differences between LoRA and DoRA, the following provides their respective simplified implementations.

Implementation

```python
class LoRAAdapter(nn.Module):
    def __init__(self, existing_layer, r, alpha):
        super().__init__()
        self.existing_layer = existing_layer
        self.r = r
        self.alpha = alpha

        # Low-rank matrices
        self.lora_A = nn.Parameter(torch.zeros(r,
              existing_layer.in_features))
        self.lora_B = nn.Parameter(torch.zeros(
              existing_layer.out_features, r))

        # Scaling factor
        self.scaling = alpha / r
        self.reset_parameters()

    def reset_parameters(self):
        nn.init.normal_(self.lora_A, mean=0.0, std=0.02)
        nn.init.zeros_(self.lora_B)

    def forward(self, x):
        original = self.existing_layer(x)
        lora_update = F.linear(x, self.lora_A)
        lora_update = F.linear(lora_update, self.lora_B)
        return original + lora_update * self.scaling
```

Listing 1: LoRA updates the weights by adding a low-rank matrix adaptation.

```python
class DoRAAdapter(nn.Module):
    def __init__(self, existing_layer, r, alpha,
          scaling_constant):
        super().__init__()
        self.existing_layer = existing_layer
        self.r = r
        self.alpha = alpha
        self.scaling_constant = scaling_constant

        # Magnitude vector
        self.m = nn.Parameter(existing_layer.weight.norm(p
            =2, dim=0, keepdim=True))

        # Low-rank matrices for directional update
        self.dora_A = nn.Parameter(torch.zeros(r,
            existing_layer.in_features))
        self.dora_B = nn.Parameter(torch.zeros(
            existing_layer.out_features, r))

        # Scaling factor
        self.scaling = alpha / r
        self.reset_parameters()

    def reset_parameters(self):
        nn.init.normal_(self.dora_A, mean=0.0, std=0.02)
        nn.init.zeros_(self.dora_B)

    def forward(self, x):
        original = self.existing_layer(x)
        direction_update = F.linear(x, self.dora_A)
        direction_update = F.linear(direction_update, self.
            dora_B)
        return self.m * (original + direction_update / self
            .scaling_constant)
```

Listing 2: DoRA decomposes the pre-trained weights into magnitude and directional components and merges them back later.

References

Abassy, M., Elozeiri, K., Aziz, A., et al. (2024). LLM-DetectAIve: a Tool for Fine-Grained Machine-Generated Text Detection. *arXiv preprint arXiv:2408.04284v2*.

Gehrmann, S., Strobelt, H., Rush, A. M. (2019). GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 111–116).

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., Chen, M.-H. (2024). DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv preprint arXiv:2402.09353*.

TheDRCat. DAIGT V2 Train Dataset. *Kaggle*. Available: https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset/data.

Thite, S. LLM Detect: AI-Generated Text Dataset. *Kaggle*. Available: https://www.kaggle.com/datasets/sunilthite/llm-detect-ai-generated-text-dataset/data.

Tolstykh, I., Tsybina, A., Yakubson, S., Gordeev, A., Dokholyan, V., Kuprashevich, M. (2024). GigaCheck: Detecting LLM-Generated Content. *arXiv preprint arXiv:2410.23728*.

Wang, A., et al. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ICLR*.

Wang, H., Li, J., Li, Z. (2024). AI-Generated Text Detection and Classification Based on BERT Deep Learning Algorithm. *arXiv preprint arXiv:2405.16422*.

Wu, J., Yang, S., Zhan, R., Yuan, Y., Wong, D. F., Chao, L. S. (2023). A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. *arXiv preprint arXiv:2310.14724.//*

Zellers, R., et al. (2019). Defending Against Neural Fake News. *NeurIPS*.

Kleinbaum, D. G., and Klein, M. (2010). Introduction to Logistic Regression. *Logistic Regression: A Self-Learning Text*, pages 1–39. Springer.

Koroteev, Mikhail V. (2021). BERT: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

Rigatti, Steven J. (2017). Random forest. *American Academy of Insurance Medicine 1700 Magnavox Way*.