CS6140 Project - Distinguishing Reality: Detecting AI-Generated Reviews in E-Commerce

Quancheng Li
Northeastern University
Seattle, USA
li.qua@northeastern.edu

Yirong Sun
Northeastern University
Seattle, USA
sun.yiro@northeastern.edu

Lizheng Wang
Northeastern University
Seattle, USA
wang.liz@northeastern.edu

Yijia Ma
Northeastern University
Seattle, USA
ma.yij@northeastern.edu

Abstract

As the prevalence of AI-generated fake reviews in e-commerce continues to rise, the integrity of product ratings and consumer trust is increasingly jeopardized. This project focuses on developing a robust machine learning model to effectively distinguish between human-written and AI-generated product reviews. Utilizing the Fake Reviews Dataset, we will explore various detection techniques, including traditional machine learning algorithms and advanced models like BERT. Our approach aims to improve the accuracy of identifying fraudulent content, thereby enhancing e-commerce platforms' ability to maintain fair competition and safeguard consumer trust. Initial evaluations will assess model performance, with the goal of addressing existing challenges in AI-generated content detection. Ultimately, this research seeks to provide insights into the evolving landscape of generative AI and its implications for online marketplaces.

1 Introduction

AI-generated fake reviews in e-commerce can distort product ratings, mislead consumers, and damage market trust. This project aims to develop a machine learning model to distinguish between human-written and AI-generated product reviews, helping e-commerce platforms filter out fraudulent content. Given the rapid rise of generative AI, this task is critical for maintaining fair competition and consumer trust in online platforms.

2 Dataset

Fake Reviews Dataset: https://www.kaggle.com/datasets/mexwell/fake-reviews-dataset/data

3 Literature Review

3.1 AI-Generated Review Detection[1]

What is the paper about? In the work done by Luo et al. (2023), a novel supervised learning approach was proposed to help e-commerce platforms effectively detect AI-generated reviews. The authors aimed to address the challenge of distinguishing between human-written and AI-generated content.
What did they do to solve the problem? They constructed three types of variables to distinguish between human-written reviews and AI-generated reviews and introduced an outlier detection method based on cumulative probability to calculate the probability of AI-generated reviews, and trained models using these features.
Conclusion: Although the proposed method can perform more accurate detections of AI-generated reviews, there are some limitations, including lack of incorporation of the behavioral features of reviewers into the AI-generated review detection framework.

3.2 Testing of Detection Tools for AI-Generated Text[2]

What is the paper about? The paper by Weber-Wulff et al. (2023) examines the general functionality of detection tools for AI-generated text and evaluates them based on accuracy and error type analysis. It seeks to answer whether existing detection tools can reliably differentiate between human-written and ChatGPT-generated text and what factors affect the detection.
What did they do to solve the problem? They tested 14 detection tools, including Check for AI, Compilatio, and Content at Scale, on datasets consisting of human-written, AI-generated, and obfuscated texts (such as machine-paraphrased or translated). They evaluated performance by measuring the accuracy of classification as (partially) true/false positives/negatives.
Conclusion: The paper exposes limitations of AI-generated text detection tools and their unsuitability for use as evidence of academic misconduct. Many results present false positives and false negatives.

3.3 Detection of Fake Customer Reviews on E-Commerce with Supervised Machine Learning[3]

What is the paper about? Patel et al. (2023) found that Naïve Bayes, Support Vector Machine, and Random Forest provide high accuracy in detecting machine-generated reviews, with Support Vector Classifier having the highest accuracy of 89.61%.
What did they do to solve the problem? They added an extra layer of feature extraction using the Bag of Words (BOW) model and TF-IDF transformer and fed the results to classification models. A probability score was calculated to assign reviews to different classes.
Conclusion: The paper compares different machine learning models in their performance on detecting machine-generated reviews

and serves as a valuable guide by offering insightful ideas on research methods and model options for our studies on fake reviews detection.

3.4 CHEAT: A Large-Scale Dataset for Detecting ChatGPT-Written Abstracts[4]

What is the paper about? In the work done by Yu et al. (2024), five text synthesis detection methods were evaluated on the CHEAT dataset, indicating that the difficulty of text synthesis detection increases with human involvement—Generation (100%), Polish (99.43% – 99.72%), and Mix (52%–87.83%).
What did they do to solve the problem? They focused on abstracts with different levels of ChatGPT intervention—Generation, Polish, and Mix—to analyze abnormal distribution patterns and provide more relevant data for developing detection methods.
Conclusion: The paper highlights the limitation of text synthesis detection algorithms by demonstrating that abstracts generated entirely by ChatGPT are detectable, while those with human guidance possess detection difficulty, especially when mixed with human-written text.

3.5 AI-Generated Text Detection and Classification Based on BERT Deep Learning Algorithm[5]

What is the paper about? In the work done by Wang et al. (2024), an efficient AI-generated text detection model was developed based on the BERT algorithm. It indicated a high average accuracy (97.71%) on a private dataset and good generalization ability on unknown data.
What did they do to solve the problem? They experimented on a private dataset and evaluated the average accuracy and loss values on the dataset in a 60/40 split, recording changes in accuracy and loss during the training process.
Conclusion: The paper utilizes the BERT algorithm to improve AI-generated text detection on a private dataset with only 1,378 labeled texts but brings challenges in reproducing the results and verifying the model's generalization ability on a larger, normally distributed dataset.

3.6 Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-Based Detection[6]

What is the paper about? In the work done by Adelani et al. (2019), the authors explore generating sentiment-preserving fake online reviews using neural language models (NLMs) such as GPT-2. They aim to show how easily accessible language models can be used to attack e-commerce review systems by generating high-quality fake reviews that are difficult to detect by both humans and machines.
What did they do to solve the problem? They used a pre-trained GPT-2 model to generate fake reviews based on existing reviews and applied a BERT-based classifier to filter out fake reviews with undesired sentiment.
Conclusion: Although their method produces fluent, realistic fake reviews, the detection accuracy by both humans and automatic systems like Grover and OpenAI GPT-2 detector remains low, indicating significant challenges in detecting AI-generated reviews.

3.7 GLTR: Statistical Detection and Visualization of Generated Text[7]

What is the paper about? In the work done by Gehrmann et al. (2019), they introduced GLTR (Giant Language model Test Room), a tool designed to help non-experts detect machine-generated text by visualizing statistical patterns that indicate whether text was generated by a language model.
What did they do to solve the problem? They applied statistical tests such as word probability, rank, and entropy, and built a visual interface to help users identify patterns in the text that differ between human-written and AI-generated content.
Conclusion: The tool improved human detection rates of fake text from 54% to 72%, demonstrating that visualizing statistical anomalies aids in detecting AI-generated content.

3.8 The Science of Detecting LLM-Generated Text[8]

What is the paper about? The paper offers a comprehensive technical overview of detection methods for large language model (LLM)-generated text, emphasizing both black-box and white-box detection strategies. It discusses the significance of detecting text generated by models like ChatGPT, particularly in the context of mitigating misuse in areas like phishing, disinformation, and academic dishonesty.
What did they do to solve the problem? They grouped existing detection methods into two categories: black-box detection, which uses external signals from the generated text to identify AI content, and white-box detection, which involves embedding watermarks or controlling the generation process directly within the LLM.
Conclusion: Although black-box detection is currently effective, it will become less reliable as LLMs improve. The future of reliable detection likely lies in white-box methods like watermarking, though these also have limitations related to text quality.

3.9 DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature[9]

What is the paper about? Mitchell et al. (2023) introduced DetectGPT, a method leveraging the gap in perturbation discrepancy for model-generated text and human-written text(Local Perturbation Discrepancy Gap Hypothesis).
What did they do to solve the problem? Minor perturbations were introduced to original passage x using a pre-trained model-T5, allowing DetectGPT to compare the log probability of x with each perturbed sample x_i. Model-generated text exhibits larger perturbation discrepancy and thresholding the discrepancy helps detect if a piece of text was generated by a model p_θ.
Conclusion: Although DetectGPT is more discriminative than existing zero-shot detection methods and does not require training a separate classifier, it is compute-intensive and it relies on access to log probability function and perturbation function and qualities of them.

4 Experimentation and Results

In this section, we discuss the experiments conducted to evaluate the performance of our models, including the Decision Tree and Random Forest classifiers. We also present the baseline comparison for accuracy and precision. The goal was to classify AI-generated reviews versus human-written ones using feature-engineered data and machine learning algorithms.

4.1 Baseline Model

To establish a point of comparison, we first implemented a baseline model using simple machine learning classifiers. The baseline performance metrics were:

- **Accuracy**: 0.8665
- **Precision**: 0.8778

This performance provides a benchmark against which we can measure the performance improvements or deficits of more complex models like Decision Tree and Random Forest.

4.2 Data Pre-processing

Before training the models, we performed several pre-processing steps to clean and prepare the data for classification. These steps ensured that both numerical and text-based features were well-represented in the models. The key pre-processing tasks were as follows:

Text Preprocessing: The *text_* column, which contained the raw reviews, was cleaned by converting all text to lowercase and removing non-alphanumeric characters. This preprocessing step ensured that the text was standardized and stripped of unnecessary symbols, allowing the models to focus on the content of the reviews.

Feature Engineering: Several new features were engineered to provide additional information to the models:

- *num_sentences*: The number of sentences in each review was extracted using the `nltk.sent_tokenize()` function. This feature provided a structural overview of the review.
- *num_characters*: The length of each review in terms of characters was computed, including spaces.
- *num_words*: The number of words in each review was calculated using `nltk.word_tokenize()`.

Handling Duplicates: Duplicate reviews, which can introduce bias and skew model performance, were identified and removed based on the *text_* column. Only the first occurrence of each unique review was retained, ensuring that repeated reviews did not affect model training.

Label Encoding: The target column *label* was encoded into a binary format, where 'CG' (likely referring to AI-generated reviews) was assigned a value of 0, and 'OR' (likely human-written reviews) was assigned a value of 1. This transformation enabled the classification models to treat the task as a binary classification problem.

Normalization: The *rating* column, representing product ratings, was normalized using min-max scaling. Normalization ensures that numerical features are scaled to a similar range, preventing any single feature from disproportionately affecting the model's performance.

One-Hot Encoding for Categorical Variables: The *category* column, which contained categorical product categories, was transformed into binary columns using one-hot encoding. Each unique category was represented as a separate feature, allowing the models to leverage category-specific information for review classification.

TF-IDF Vectorization: The review text was vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) technique, capturing the importance of words in each review relative to the entire corpus. This step converted the textual data into numerical features, allowing the models to identify important words or patterns in the reviews. The number of TF-IDF features was limited to 3000 to reduce dimensionality and computational complexity.

Combining Features: The final feature set was created by combining the TF-IDF matrix with the numerical and one-hot encoded features, such as the number of sentences, words, characters, ratings, and categories. This ensured that both textual and non-textual data were included in the model training process.

4.3 Decision Tree Classifier

We implemented a Decision Tree classifier as the first model for comparison. Decision Trees are simple to interpret and can effectively capture nonlinear relationships between features.

Results - Decision Tree:

Class	Precision	Recall	F1-score	Support
0	0.76	0.78	0.77	4009
1	0.78	0.76	0.77	4073
Accuracy	0.7690			8082
Macro avg	0.77	0.77	0.77	8082
Weighted avg	0.77	0.77	0.77	8082

Training set score	1.0000
Test set score	0.7690

Table 1: Best Performance of Decision Tree Classifier

While the Decision Tree performed reasonably well with a test accuracy of 76.90%, its performance did not match that of the baseline model. One of the key drawbacks of Decision Trees is their tendency to overfit, as evidenced by the perfect training accuracy of 100%. This overfitting could potentially limit the model's ability to generalize to new, unseen data.

The confusion matrix and Receiver Operating Characteristic (ROC) curve in Figure 1 and Figure 2, respectively, show the model's classification distribution and its ability to distinguish between positive and negative classes.

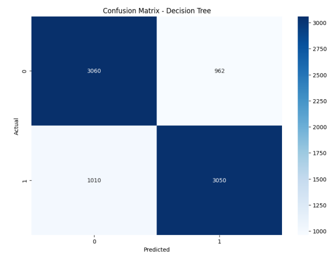

Figure 1: Confusion Matrix - Decision Tree

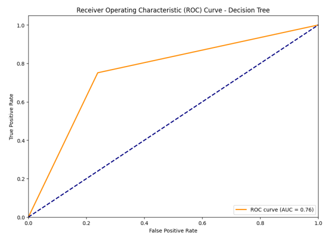

Figure 2: ROC Curve - Decision Tree

Furthermore, we evaluated the feature importance in the Decision Tree model, shown in Figure 3. The most important features were related to word counts, the presence of specific words, and sentence counts. This suggests that the structural properties of the reviews are crucial in distinguishing between AI-generated and human-written reviews.

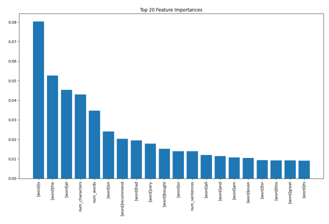

Figure 3: Top 20 Feature Importances - Decision Tree

4.4 Random Forest Classifier

Given the limitations of the Decision Tree model, we next implemented a Random Forest classifier, which is an ensemble learning method designed to reduce overfitting by averaging the results of multiple decision trees.

Results - Random Forest:

Class	Precision	Recall	F1-score	Support
0	0.87	0.90	0.89	4036
1	0.90	0.87	0.88	4047
Accuracy	0.8847			8083
Macro avg	0.89	0.88	0.88	8083
Weighted avg	0.89	0.88	0.88	8083

Training set score	0.9999
Test set score	0.8847

Table 2: Best Performance of Random Forest Classifier

The Random Forest model significantly outperformed the Decision Tree, achieving an accuracy of 88.47% and a precision of 89%. As expected, the ensemble approach improved the generalization of the model, as indicated by its lower overfitting and better test accuracy compared to the Decision Tree.

The confusion matrix and ROC curve for the Random Forest are shown in Figures 4 and 5, respectively. The ROC curve shows that the model has a much better ability to distinguish between positive and negative classes compared to the Decision Tree.

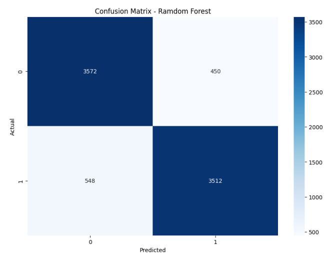

Figure 4: Confusion Matrix - Random Forest

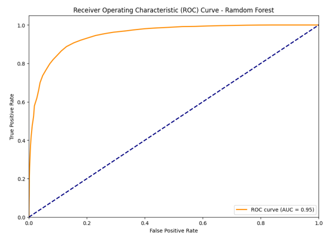

Figure 5: ROC Curve - Random Forest

The Random Forest's feature importance, shown in Figure 6, indicated similar insights to the Decision Tree. The most critical features were related to word counts and specific word frequencies, but the Random Forest also accounted for more intricate interactions between the features, improving its classification performance.

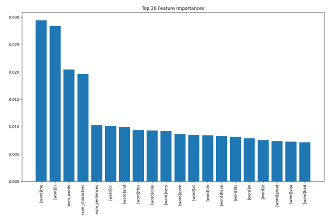

Figure 6: Top 20 Feature Importances - Random Forest

4.5 Conclusion(4.1-4.4)

Comparison with Baseline: When compared to the baseline model, the Random Forest classifier improved the accuracy and precision, achieving an accuracy of 88.47% and precision of 89%, compared to the baseline's 86.65% accuracy and 87.78% precision. The Decision Tree, although less complex, achieved lower accuracy and precision compared to both the baseline and Random Forest models.

In conclusion, the Random Forest model is the best-performing model in our experiments, outperforming both the baseline and the Decision Tree classifiers. While the Decision Tree provided useful insights into feature importance, its tendency to overfit limited its practical utility. The Random Forest's ability to generalize across different feature sets makes it the most suitable model for detecting AI-generated reviews in this context.

The next steps will involve exploring more advanced models, such as BERT and GPT, to see if deep learning techniques can further improve the accuracy and generalization of AI-generated review detection.

4.6 BERT with Fully Connected Layer

To improve the detection of AI-generated reviews, we implemented a BERT-based model architecture with an additional fully connected layer. This design leverages BERT's powerful contextual embeddings and combines them with other features like category and rating, enabling the model to better capture complex linguistic patterns and contextual relationships within the text.

Training Results (Using Fold 5 as an Example): Among the five cross-validation folds, Fold 5 achieved the highest validation accuracy, reaching 99.93%. Here is a detailed analysis of the training process and results from Fold 5:

- **Loss Trends**: The training and validation loss trends show that the training loss steadily decreased across epochs, while the validation loss initially rose before decreasing in the final epoch, indicating improved generalization. (See Figure 7 left).
- **Validation Accuracy**: The validation accuracy showed a significant upward trend, peaking in the last epoch, which highlights the model's generalization ability (Figure 7 right).
- **Confusion Matrix Analysis**: The confusion matrix for Fold 5 demonstrated near-perfect classification, with only 3 misclassifications in each class, reflecting balanced performance (Figure 8).
- **Feature Importance**: The feature importance analysis showed similar values across BERT embeddings, category, rating, and text-based metrics, indicating the model effectively utilized a diverse set of features (Figure 9).

The results from Fold 5 highlight the effectiveness of combining BERT with a fully connected layer, achieving a 99.93% validation accuracy, which underscores its strong capability in detecting AI-generated content.

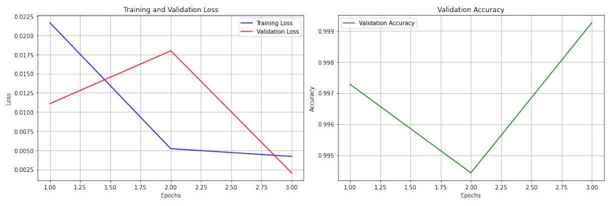

Figure 7: Fold 5 Training and Validation Loss (left) and Validation Accuracy (right).

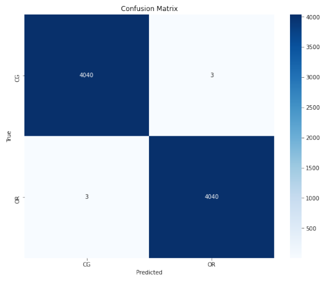

Figure 8: Confusion Matrix for Fold 5, showing near-perfect classification performance.

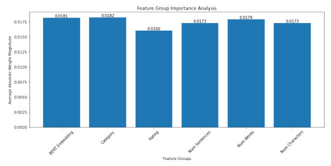

Figure 9: Feature Importance Analysis for Fold 5, showing contributions from BERT embeddings, category, rating, and text-based metrics.

Summary: Based on the excellent performance in Fold 5, our BERT model achieved an average accuracy of 98.77% across all folds, with a standard deviation of 1.05%. This demonstrates the model's strong generalization ability and consistency across different data splits. This BERT-based architecture offers a highly effective and reliable solution for detecting review authenticity in e-commerce platforms.

4.7 Zero-Shot Fake Review Detection using Probability Curvature

Inspired by the work done by Mitchell et al. (2023), we migrate the DetectGPT architecture. This design leverages the demonstrated gap in perturbation discrepancies of machine-generated text and human-written text, which approximates a measure of the local curvature of the log probability function near the candidate passage.

In this implementation, GPT-2 is used as the base model to calculate the log-likelihood of input texts, and a separate mask-filling model, T5-large, is used to generate perturbations. For each input, perturbed texts are created by replacing randomly chosen spans of text with 10 tokens, a value we set but can be configured as needed, which are then filled with plausible alternatives by the mask-filling model (T5-large). This process generates a perturbed text distribution for which the log-likelihoods are compared against the original text.

The implementation is structured to:

- *Text Perturbation*: The perturb_texts function applies random masking to input texts and generates plausible substitutions using the mask-filling model(T5-large).
- *Log-Likelihood Calculation*: The log-likelihoods of original text and perturbed texts are computed using the base model.

- *Score Derivation*: Depending on the criterion, direct or normalized log-likelihood differences are calculated and used to compute metrics like ROC-AUC and Precision-Recall AUC. (See Figure 11, Figure 12, Figure 13, and Figure 14).

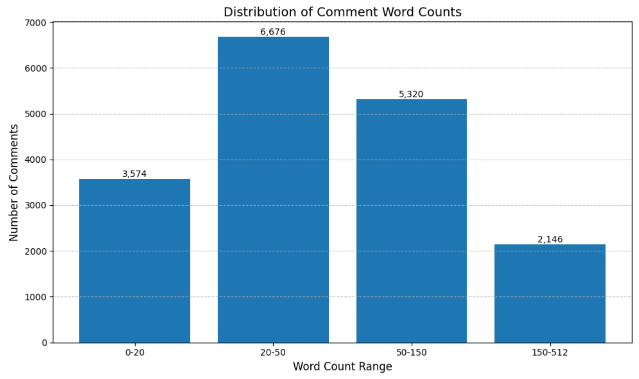

Figure 10: Word Count Distribution of Comments

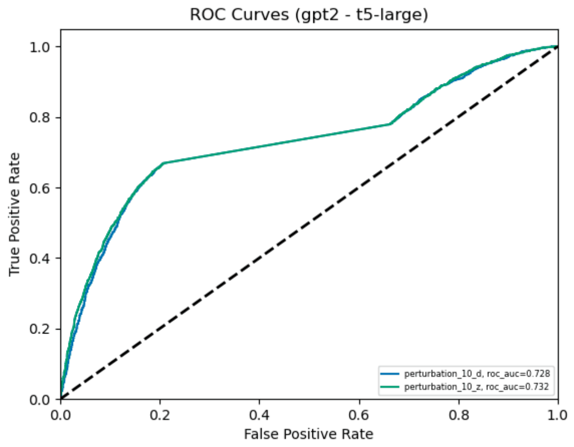

Figure 11: ROC Curve for Reviews with Word Count Between

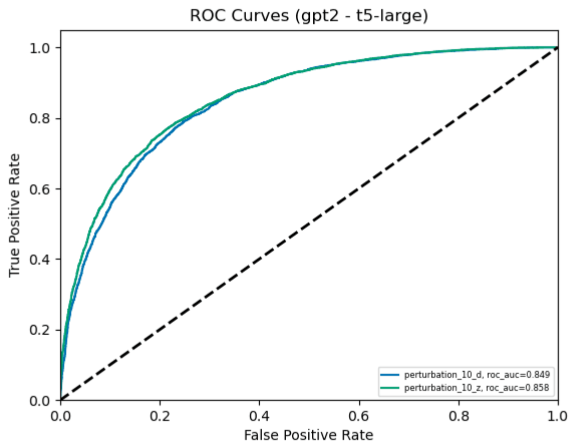

Figure 12: ROC Curve for Reviews with Word Count Between 20 and 50

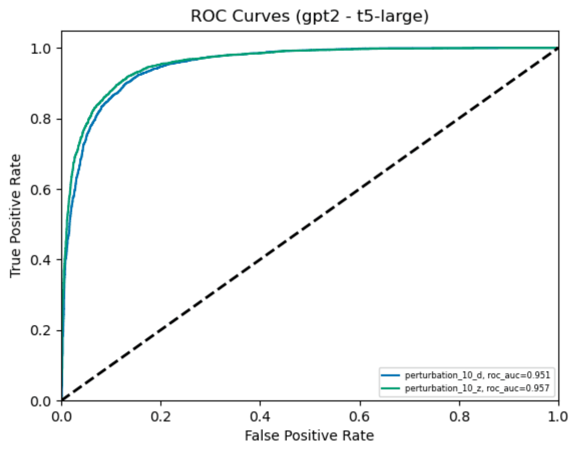

Figure 13: ROC Curve for Reviews with Word Count Between 50 and 150

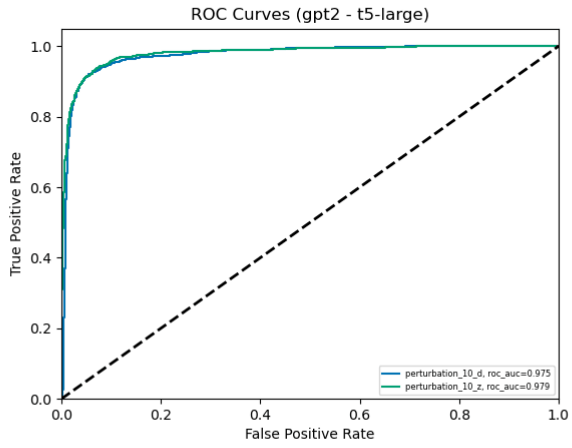

Figure 14: ROC Curve for Reviews with Word Count Between 150 and 512

Summary: Figure 10 illustrates the word count distribution of comments and it further impacts the quality of text perturbation. This can be attributed to the perturbation process, as described in prior implementations, which operates at the sentence level. For reviews with lower word counts, a larger portion of the semantic information may be altered or lost, leading to reduced performance on shorter reviews. This observation aligns with the trends seen in the ROC curves presented in Figure 11, Figure 12, Figure 13, and Figure 14.

DetectGPT with a z-score criteria achieved a ROC-AUC of 0.979. This demonstrates the underlying Perturbation Discrepancy Gap Hypothesis of zero-shot method and highlights the inherent differences between human-written and machine-generated text, showing the generalization ability to detect machine-generated text in other fields besides e-commerce platforms.

Limitations: DetectGPT operates under several key assumptions, which are outlined as follows:

- **Log Probability Function**: As a probability-based method, DetectGPT relies on the availability of a mechanism to evaluate the log probabilities of the target model. This can be achieved either through a built-in log probability function or access to the model's API, though the latter often increases operational costs.
- **Perturbation Function**: The effectiveness of DetectGPT heavily depends on the quality of the perturbation function. A well-designed perturbation function is essential to represent the space of meaningful rephrases and improve the accuracy of the curvature estimate.
- **Computational Cost**: DetectGPT is computationally intensive as it requires sampling and scoring a set of perturbations for each candidate passage, rather than evaluating just the candidate passage. Developing a more efficient perturbation function or optimizing the curvature approximation process could help reduce these computational demands.

5 Project Plan and Milestones

- **Week 3**: Project Proposal Submission—Determine the target and dataset of the project.
- **Week 4**: Data Preparation & Feature Engineering—Explore and preprocess the dataset.
- **Week 5**: Model Selection & Training (Baseline)—Implement and evaluate baseline models.
- **Week 6**: Project Update 1—Prepare and submit slide deck, Python code, and report.
- **Week 7**: Implement Advanced Models—Implement and tune advanced models (e.g., BERT, GPT).
- **Week 8**: Comparative Analysis—Conduct comparative analysis between models.
- **Weeks 9–10**: Project Update 2—Prepare and deliver system demo.
- **Weeks 11–12**: Feedback-Based Improvements—Analyze feedback and implement improvements.
- **Weeks 13–14**: Final Project Submission—Complete and submit all final deliverables (slide deck, code, report, and peer review).

Acknowledgments

We would like to thank our professor and teaching assistants for their guidance throughout this project.

References

[1] Luo, J., Nan, G., Li, D., & Tan, Y. (2023). *AI-Generated Review Detection.* SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4610727
[2] Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S., Foltýnek, T., Guerrero-Dib, J., Popoola, O., Šigut, P., & Waddington, L. (2023). *Testing of detection tools for AI-generated text.* International Journal for Educational Integrity, 19(1). https://doi.org/10.1007/s40979-023-00146-z
[3] Patel, K., & Daneshyari, M. D. (2023). *Detection of fake customer reviews on e-commerce with supervised machine learning.* International Journal of Computer Programming and Database Management, 4(1), 85–89. https://doi.org/10.33545/27076636.2023.v4.i1a.83
[4] Yu, P., Chen, J., Feng, X., & Xia, Z. (2024). *CHEAT: A large-scale dataset for detecting ChatGPT-written abstracts.* arXiv preprint arXiv:2304.12008. http://arxiv.org/abs/2304.12008
[5] Wang, H., Li, J., & Li, Z. (2024). *AI-generated text detection and classification based on BERT deep learning algorithm.* Theoretical and Natural Science, 39, 312–317.
[6] Adelani, D. I., Mai, H. T., Fang, F., Nguyen, H. H., Yamagishi, J., & Echizen, I. (2019). *Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection.* In International Conference on Advanced Information Networking and Applications (pp. 1341–1354).
[7] Gehrmann, S., Strobelt, H., & Rush, A. M. (2019). *GLTR: Statistical detection and visualization of generated text.* arXiv preprint arXiv:1906.04043.
[8] Tang, R., Chuang, Y.-N., & Hu, X. (2023). *The Science of Detecting LLM-Generated Text.* arXiv preprint arXiv:2309.03481.
[9] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., & Finn, C. (2023, July). *Detectgpt: Zero-shot machine-generated text detection using probability curvature.* International Conference on Machine Learning (pp. 24950-24962). PMLR.