Advanced Feature Engineering for AI Text Detection

Kasper Thomas Gartside Knudsen - kakn@itu.dk

KISPECI1SE, Computer Science, IT-University of Copenhagen
Supervised by Associate Professor Dr. Christian Hardmeier

Abstract

The differentiation between AI-generated and human-written texts is crucial in maintaining academic integrity and ensuring content authenticity. This thesis explores a feature-based classification approach to detect AI-generated text, integrating established NLP text classification features with novel attributes designed to capture the nuances of AI-generated content. By employing SHAP (SHapley Additive exPlanations) to interpret model predictions, we identify the most influential features in distinguishing AI-generated from human-written text. Our results demonstrate the effectiveness of new features such as unigram entropy, burstiness, and average dependent clauses. Additionally, we evaluate the robustness of these detection models against texts generated with evasive prompts using an unseen LLM (Llama-3-8B). Comparative analysis with fine-tuned LLM classifiers (DistilBERT) and TF-IDF based models highlights the strengths and limitations of each approach. The study also discusses the ethical implications of AI text detection in academic settings and proposes potential applications in content moderation and beyond. The findings underscore the importance of continuous feature engineering and the development of advanced LLM models to keep pace with evolving AI text generation techniques.

Introduction

The capabilities of modern large language models (LLMs) in generating human-like text introduce the risk of misuse in academia, phishing, and disinformation. Consequently, there is growing interest in academia and industry to research methods for detecting AI-generated text and to improve our understanding of its underlying patterns.

This thesis explores a feature-based classification approach to differentiate between AI-generated and human-written texts. By integrating traditional NLP text classification features with novel attributes specifically designed to detect the nuances of AI-generated content, we aim to develop a robust detection model. Specifically, we aim to enhance feature collections by incorporating measures of complexity and repetition, as we expect these traits to be more prevalent in AI-generated texts.

To provide a fair evaluation, we generate text from an unseen-to-the-classifier large language models (LLM) both evasively and normally, testing the robustness and transferability of the model. Additionally, we offer a comparative analysis of our approach by implementing baselines that replicate established feature-based methods, as well as fine-tuned and zero-shot LLM-based approaches. We hypothesize that our research into new and relevant features will improve upon these baselines, underscoring the importance of continued research to keep pace with evolving LLM methods.

Through comprehensive experiments and evaluations, we aim to enhance the understanding and effectiveness of AI text detectors, contributing to their practical applications in real-world scenarios. Additionally, we use this intuition, along with related literature, to judge whether it is an appropriate tool to judge academic misconduct and plagiarism. All code used in this research is accessible on the project's GitHub repository.[1]

Literature Review

This paper builds on established approaches in AI text detection, aiming to assess the effectiveness of extensive feature engineering compared to more sophisticated methodologies. Furthermore, we discuss the relevance and usability of AI text detectors in both academic and real-world contexts, drawing from comparative studies of popular AI-detection models and research on the ethics of this field. Lastly, we provide a brief overview of promising and cutting-edge alternative methodologies in this field that were not feasible to implement in this thesis.

Classification of Human- and AI-Generated Texts: Investigating Features for ChatGPT

The study [1] presents a comprehensive methodology for distinguishing between human-written and AI-generated texts, employing a feature-based approach that leverages ensemble learning models such as XGBoost and Random Forest. This research integrates a wide range of NLP text classification features, chosen for their proven success in AI text detection across various studies. Additionally, the paper introduces some novel features specifically designed to address

[1] `https://github.itu.dk/kakn/mscthesis`

the nuances of AI-generated content. These features can be categorized as follows:

- **Perplexity-Based Features:** Measures the mean and maximum perplexity of sentences, showing higher predictability within AI texts.
- **Semantic Features:** Involves sentiment polarity and subjectivity/objectivity levels, with AI texts showing higher subjectivity.
- **List Lookup Features:** These include counts of words or characters such as stop words, special characters, and discourse markers, revealing more varied usage in human texts. Additionally, the absolute and relative repetitions of the article's title are tracked, as AI-generated texts often repeat title keywords.
- **Error-Based Features:** AI texts have fewer spelling and grammar errors than human texts. Counts of such errors and multiple blanks are detected, with human texts showing more errors.
- **Document Features:** This includes 20 various combinations of relative and absolute counts, means, and standard deviations of features such as words, sentences, punctuation and others, showing less variability in AI text.
- **Readability Features:** Includes the Flesch Reading Ease score and Flesch-Kincaid Grade Level. AI-generated texts generally show higher grade levels, indicating higher complexity.
- **AI Feedback Features:** Involves asking ChatGPT about the text's origin and assigning binary predictions based on responses.
- **Text Vector Features:** Analyzes text semantics through TF-IDF and Sentence-BERT. The average distances of Sentence-BERT vectors help detect repetitive patterns in AI-generated texts.

A unique corpus of texts was created by web scraping Wikipedia and paraphrasing this information with ChatGPT, resulting in *human* and *AI* texts. The research found that a Random Forest model performed best, achieving 98% accuracy when using both novel and established features, slightly improving upon the use of established features alone. The study further validates the robustness of this model with AI texts instructed not to sound like AI, showing a significant decrease in accuracy and mixed results regarding the effectiveness of the new features in this context.

ChatGPT Generated Text Detection

This study [2] conducts a comparative analysis of feature extraction strategies for detecting ChatGPT-generated texts using the XGBoost algorithm, contrasting TF-IDF vectorization with an extensive collection of hand-crafted features.

The TF-IDF approach captures word importance through frequency and inverse document frequency, while the hand-crafted features encompass a broad range of lexical, syntactic, and semantic characteristics. Although the feature list includes 244 features, it primarily consists of variations of a smaller set of unique features repeated for different n-grams.

Experimental results show that both approaches effectively detect AI-generated texts. The TF-IDF model achieved 90% accuracy, while the hand-crafted feature model attained 96%. These findings highlight the potential of intensive feature models and demonstrate the utility and cost-effectiveness of TF-IDF in capturing essential word-level information, despite its simplicity, making it a strong baseline for feature extraction methods.

The study also employs SHAP (SHapley Additive exPlanations) visualizations to interpret model predictions, providing clear insights into feature importance and the characteristics of AI-generated text. This is particularly useful for understanding the TF-IDF model's decisions and identifying words that are consistently more prevalent in AI texts.

Testing of Detection Tools for AI-Generated Text

This paper [3] conducts a comprehensive analysis of AI-generated text detection tools, focusing on evaluating the tools' capability to differentiate between human-written and AI-generated (specifically ChatGPT-generated) texts, as well as examining the impact of machine translation and content obfuscation techniques on detection accuracy. Twelve publicly available tools and two commercial systems (Turnitin and PlagiarismCheck) were tested.

The paper concludes that current detection tools have serious limitations, especially in detecting false positives and false negatives, and states they are unsuitable for use as evidence of academic misconduct. Turnitin further validates these findings on their website,[2] where they state:

> *"While Turnitin has confidence in its model, Turnitin does not make a determination of misconduct, rather it provides data for the educators to make an informed decision based on their academic and institutional policies. Hence, we must emphasize that the percentage on the AI writing indicator should not be used as the sole basis for action or a definitive grading measure by instructors."*

The paper also finds that the detectors are too easy to trick with use of paraphrasing tools or machine translation. Alternative methods of AI use prevention are suggested, including a larger focus on preventive measures and academic assessment strategies.

[2] `https://www.turnitin.com/products/features/ai-writing-detection/`

How Reliable Are AI-Generated-Text Detectors? An Assessment Framework Using Evasive Soft Prompts

The study [4] explores the effectiveness of AI-generated text detectors in the face of sophisticated evasion techniques. The authors developed a framework called EScaPe (Evasive Soft Prompt) to generate text that can deceive these detectors. A soft prompt is a set of learned embeddings integrated directly within a language model's architecture to guide it in producing specific types of text. This contrasts with traditional "hard prompts," which utilize carefully designed textual instructions when the model is generating or predicting new text. EScaPe introduces a two-step process involving the learning of an evasive soft prompt for a specific frozen pre-trained language model (PLM), followed by the transfer of this prompt to other frozen target PLMs.

The first step, evasive soft prompt learning, focuses on tuning the soft prompt so that it can guide PLMs to generate text that is classified as "human-written" by the detector. The second step, evasive soft prompt transfer, involves transferring the trained evasive soft prompt to the semantic space of the target PLM, thereby enabling its application across different models. This framework represents a more advanced method of evaluating the reliability of AI text detectors and underscores the importance of continuous improvement in detection methods in the era of advanced AI.

SeqXGPT: Sentence-Level AI-Generated Text Detection

SeqXGPT [5] stands out as an AI-text detection approach by analyzing AI-generated text on a sentence level, by leveraging log probability lists derived from open-source Large Language Models (LLMs) as a single feature in detecting AI-generated text. By processing the log probabilities as waveforms—a method inspired by speech processing—the implementation demonstrates particular success with sentence-level AI text detection. AI-generated text may often not comprise an entire document, making the SeqXGPT sentence-level approach more realistic in a real-world setting than a document-level methodology. Additionally, the research demonstrates that this single feature is effective in the field of AI text detection, achieving an accuracy of 95.7%.

A Simple yet Efficient Ensemble Approach for AI-generated Text Detection

The paper [6] focuses on detecting AI-generated text using Large Language Models (LLMs) and transformer-based models. The authors propose a simple ensemble approach that fine-tunes and utilizes predictions from only two LLMs, namely RoBERTa base OpenAI detector and XLM-RoBERTa NLI. This contrasts with previous methods that rely on more models or perplexity-based measures. The "Short Ensemble" model delivers performance comparable to more complex ensembles, showing significant improvements across various benchmark datasets.

Additionally, the study explores replacing GPT-generated data with open LLM data from models like LLaMA2, Falcon, and Mosaic Pretrained Transformers (MPT). Results show these alternatives perform comparably or better, avoiding commercial restrictions associated with GPT data. The model's robustness is tested through zero-shot generalization on essays from native and non-native English speakers. Despite some limitations, it outperformed several existing AI-generated text detection tools, especially in identifying essays by native speakers, highlighting its potential for broader applications in detecting AI-generated content.

Methodology

Dataset Description

The dataset utilized for this thesis [7] is a large-scale collection specifically designed for AI text detection tasks, with a focus on long-form texts and essays, allowing for the identification of patterns that may not be evident in shorter texts. This aligns with guidelines from OpenAI's GPT-2 Output Detector [8], which suggests that detection reliability improves after approximately 50 tokens. The dataset was sourced from Hugging Face and selected for its extensive variety of human and AI text samples, ensuring that AI text detectors can generalize effectively across different LLMs and topic domains. Human texts include stories and essays from Reddit prompts, question-and-answer responses from various sources, and academic essays from IvyPanda, while AI texts were generated with various LLMs, using different prompting techniques such as temperature sampling, top-K truncation, pairwise comparisons, and instructional prompts. The distributions of AI-generated text within the dataset are as follows:

- **GPT-2:** 63.34%
- **GPT-3:** 18.03%
- **GPT-J:** 10.72%
- **ChatGPT:** 7.92%

This distribution emphasizes open-source LLM data, and facilitates the development of classifiers that can potentially generalize well to other LLM domains.

The dataset comprises ~1.39 million rows, with 26.6% of the entries being AI-generated text. To manage the dataset size and ensure an even distribution of classes, we sample the human-written text to match the size and maximum length of the AI-generated text subset. This approach simplifies the research process and allows for a more reliable evaluation of the classifier's performance. After processing, the final dataset contains approximately 680,000 rows. Each row includes:

- **id:** A unique identifier for each row of data
- **text:** The text itself, up to 1024 characters in length
- **ai_generated:** A binary label indicating whether the text is AI-generated (1) or not (0)

Feature-based AI Text Detectors

Feature-based approaches have long been valuable for text analysis and remain highly relevant with the insights they provide into AI-generated text. By continually identifying new patterns in AI-generated text, we can further enhance classification accuracy and robustness. To this end, we define a comprehensive feature extraction approach, informed by established methods and new intuitions about the characteristics of AI-generated text.

Hand-crafted Features We replicate an extensive collection of features used for AI text detection [1] as a baseline for comparison with our extended model, which incorporates new features aligned with our hypotheses on AI text patterns. The baseline approach contains 36 hand-crafted features, covering perplexity-based measures, semantic attributes like sentiment polarity and subjectivity, various list lookup counts, error detection metrics, document statistics, readability scores, ChatGPT predictions, and text vector analyses using TF-IDF and Sentence-BERT. All baseline features were reproduced except for article title repetitions, which are not applicable to this dataset, and ChatGPT predictions, which were replaced with an open-source model due to the large amount of training data.

To improve on this extensive feature set, we identify features aligned with specific assumptions about AI-generated text, rather than traditional text classification features sufficiently covered by the baseline approach. This way, we leverage the specificity of a feature-based approach to capture nuances unique to AI-generated text.

We hypothesize that AI-generated texts are more complex due to the sophisticated and diverse nature of their training data, which may include research papers and other high-level materials. This enables AI models to produce complex outputs with a higher vocabulary and a writing style that resembles scholarly work, making it more intricate than the average person's writing. Additionally, we expect AI-generated text to be more repetitive than human text because AI models probabilistically generate text, potentially leading to the reiteration of phrases due to patterns in their training data. Unlike humans, AI models may not prioritize avoiding redundancy. Finally, we expect AI-generated text to frequently use list items because AI models are trained to structure information clearly, reflecting common formatting patterns found in their training data.

Based on these hypotheses, we develop specific features to detect AI-generated text:

1. AI-generated text often appears more complex than human text, with an unusual level of detail and sophisticated language.
   (a) **Average Number of Dependent Clauses**: Measures the mean number of dependent clauses per sentence, indicating higher complexity.
   (b) **Count of Passive Voice**: Counts passive constructions within the text, potentially more frequent in AI-generated text to create a formal tone.
   (c) **Average Syntactic Tree Depth**: Calculates the average depth of syntactic trees by traversing each sentence's parse tree and averaging the depths across all sentences.

$$\text{ASTD} = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{1}$$

   Where N is the total number of sentences, and d_i is the depth of the syntactic tree for the i-th sentence.

2. AI text tends to repeat information, sometimes redundantly:
   (a) **N-gram Entropy**: Measures the entropy of unigram, bigram, and trigram distributions, with higher entropy indicating more variety. In AI texts, we expect higher unigram entropy, indicating complexity, and higher bigram and trigram entropy, indicating repetition.

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{2}$$

   Where X is the set of n-grams (unigrams, bigrams, or trigrams) and $p(x)$ is the probability of the n-gram x. The probability $p(x)$ is calculated as the frequency of the n-gram x divided by the total number of n-grams in the text.

(b) **Burstiness**: Measures the variability in word frequencies, indicating how certain words cluster together, which is often seen in AI-generated text. High burstiness suggests that specific words appear frequently in close succession, reflecting repetitive patterns, while low burstiness indicates more evenly distributed word usage [9].

$$B = \frac{\sigma_w}{\mu_w} \tag{3}$$

Where σ_w is the standard deviation of word frequencies and μ_w is the mean word frequency.

3. AI text frequently uses lists:
   (a) **Count of List Items**: Counts the number of list items in the text, identifying frequent use of bulleted, hyphenated, and numbered lists.

Finally, we define our *extended* collection of features as follows. These features are standardized before being used for training our models:

Group	Feature
Sentiment	Polarity
	Subjectivity
Semantics	Stop Word Count
	Special Character Count
	Punctuation Count
	Quotation Count
	Uppercase Words Relative
	Personal Pronoun Count
	Personal Pronoun Relative
	POS Per Sentence Mean
	Discourse Marker Count
Structure	Character Count
	Word Count
	Sentence Count
	Paragraph Count
	Words Per Sentence Mean
	Words Per Sentence Std
	Words Per Paragraph Mean
	Words Per Paragraph Std
	Sentences Per Paragraph Mean
	Sentences Per Paragraph Std
	List Item Count
Lexicon	Unique Word Count
	Unique Words Per Sentence Mean
	Unique Words Per Sentence Std
	Unique Words Relative
	Unigram Entropy
Readability	Flesch Reading Ease
	Flesch Kincaid Grade
Complexity	Passive Voice Count
	Syntactic Tree Depth Per Sentence Mean
	Dependent Clauses Per Sentence Mean
Repetition	Burstiness
	Bigram Entropy
	Trigram Entropy
Errors	Error Count
	Multi Blank Count
Text Vectors	500-dim TF-IDF Vector (Uni/Bigram)
	Sentence-BERT Vector
	Average Sentence-BERT Distance
LLM-based	Max Perplexity
	Mean Perplexity
	Zero-Shot LLM Predictions

TF-IDF Given the complexity of other approaches in this thesis, it is worthwhile to evaluate a simpler approach for AI text detection, namely relying solely on TF-IDF (Term Frequency-Inverse Document Frequency) as a feature in a classifier, a method proven effective in various studies on AI text detection [2] [10]. TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a corpus. The TF-IDF value increases proportionally with the number of times a word appears in the document but is offset by the frequency of the word in the corpus, helping to account for the fact that some words are generally more common than others.

Additionally, this methodology may provide insights into the terminology that is more frequently used in AI-generated text. A recent study by Dr. Jeremy Nguyen discovered that the usage of the word "delve" in PubMed articles has surged by around 400 percent since late 2022, a trend that aligns with the emergence of ChatGPT [11]. By visualizing the top features of our TF-IDF model with SHAP [12], a method for interpreting the impact of each feature on predictions, we may be able to identify other words that are indicative of LLMs.

To establish a reliable baseline, we replicate a prior study that reports an accuracy of 90% [2]. This study uses an XGBoost model for AI text classification, with a TF-IDF matrix as input data. The preprocessing steps include tokenizing, lowercasing, stemming, and removing stop words. In our implementation, we only lowercase to preserve potential AI-specific patterns. The TF-IDF vectorizer uses unigrams and bigrams and limits the number of features to the top 500, focusing on the most frequent features and potentially preventing overfitting.

We will further evaluate this method on texts generated by an unseen large language model (LLM) to test the hypothesis that the TF-IDF method may generalize well to outputs from different models, assuming semantic consistency across LLMs.

Feature-based Model Architecture For the handcrafted and TF-IDF feature approaches, we select XGBoost as the model architecture, due to its success and popularity in AI-generated text detection [1] [6] and its optimal performance in classification tasks on tabular data [13] [14]. XGBoost is a powerful gradient boosting algorithm that adds decision trees sequentially, improving accuracy by iteratively correcting the errors of previous trees [15]. This process focuses on difficult samples, making it effective for classification and regression tasks. To optimize this model per approach, we define an extensive hyperparameter search range, with initial experiments showing success using high n_estimators combined with low learning rates and balanced max_depth to avoid excessively long training times. Given the computational expense, we use randomized search with 3-fold

cross-validation and 20 iterations, despite some recommendations for 5 or 10-fold cross-validation [16]. The parameter grid is as follows:

- **max_depth**: [3, 4, 5, 6]
  Defines the maximum depth of each decision tree. Higher values increase complexity and risk of overfitting. A range of 3 to 6 balances training time and complexity.
- **learning_rate**: [0.01, 0.05, 0.1]
  Controls the step size during the boosting process. Lower values make the model more robust to overfitting but take longer to converge.
- **n_estimators**: [2000, 2500, 3000, 3500, 4000]
  The number of decision trees to be added sequentially. Higher numbers can improve performance but increase computational cost. Initial experimentation found the chosen range to be effective.
- **colsample_bytree**: [0.6, 0.8]
  The fraction of features to be used per tree. Lower values introduce randomness to prevent overfitting.
- **min_child_weight**: [1, 3, 5]
  Minimum total weight needed in a child node to make a split in a decision tree. Higher values make the model more conservative and help prevent overfitting.
- **reg_alpha**: [0, 0.01, 0.1]
  L1 regularization term. Encourages sparsity in the model.
- **reg_lambda**: [0.01, 0.1]
  L2 regularization term. Helps prevent overfitting.

We perform this hyperparameter search for our baseline and extended hand-crafted feature sets and our TF-IDF feature set, ensuring a fair comparison across these approaches. Once we have our tuned models, we use SHAP [12] to interpret and visualize the impact of features on model predictions.

SHAP works by estimating the contribution of each feature to a prediction by comparing the difference in the model's output when the feature is included versus when it is excluded, averaged over all possible feature combinations. This method provides a consistent measure of feature importance, helping us understand how each feature influences the model's decision. We will visualize SHAP on our TF-IDF and extended feature set XGBoost models to identify the terms and features that contribute the most to the model's predictions.

LLM-based AI Text Detectors

In this section, we introduce the experimental setup of the LLM models used to classify AI-generated text, contrasting the traditional approach of feature engineering with the advanced capabilities of modern LLMs. Two approaches have been implemented in this study that utilize LLMs:

Fine-tuned LLM Classifier In contrast to feature-based and zero-shot LLM approaches, we implemented a fine-tuned LLM text classifier following Hugging Face guidelines on text classification [17]. We hypothesize that a generic pre-trained model may better generalize across unseen LLM texts than our feature-based model due to its broader contextual understanding, leading to more consistent performance. The Hugging Face guide provides detailed instructions on using its libraries and PyTorch to create custom text classifiers, covering steps such as preprocessing, training, evaluation, and inference.

We selected DistilBERT, a lightweight LLM with 66 million parameters [18], which offers comparable performance in text classification to larger models like BERT with 110 million parameters, but with faster training times and smaller model sizes [19]. BERT (Bidirectional Encoder Representations from Transformers) is a language model that uses the Transformer architecture to understand context by considering both left and right words in all layers [20]. It is trained on masked language modeling (predicting hidden words) and next sentence prediction (predicting the order of two sentences). DistilBERT is a smaller, faster version of BERT, created using knowledge distillation. It retains 97% of BERT's language understanding while being 40% smaller. Its potential in this domain is illustrated by being among the top models in a Kaggle competition on AI text detection [21] in terms of popularity and accuracy, despite its smaller size. Furthermore, DistilBERT's small size allows it to run without high-performance computing resources, underscoring its potential as a commercial AI text detection solution where GPU costs may be a consideration.

The model was initialized with the *distilbert-base-uncased* pre-trained model from Hugging Face, adapting the tokenizer and model for binary text classification tasks (*human* vs. *AI*). The tokenizer was configured to tokenize, pad, and truncate texts to a maximum length of 512 tokens. Given our maximum text length of 1024 characters, this token limit is adequate.

Text preprocessing involved tokenization, lowercasing, stemming, and removal of stop words. The dataset was split into training, validation, and test sets to ensure robust evaluation. A data collator dynamically padded inputs to the longest sequence in each batch, optimizing training efficiency.

Hyperparameter optimization was conducted using Optuna, an open-source framework for efficient

and flexible tuning [22]. Optuna's advanced algorithms, dynamic search space adjustment, and pruning strategies make it ideal for optimizing complex and resource-intensive models like DistilBERT. We optimized standard parameters such as learning rate, number of training epochs, batch size, weight decay (for regularization), and warmup steps (initial learning rate ramp-up). The search was based on validation set performance, and we limited the number of trials to 10 for efficiency, balancing thoroughness and resource constraints, as training fine-tuned LLMs takes a very long time. Training employed mixed precision to further speed up training and reduce memory usage.

Zero-shot LLM classifier Zero-shot classification is an approach in which a model is tasked with classifying data into categories it has not previously encountered during its training phase. Previous studies have reported accuracies of 62% for zero-shot classification in AI text detection when used as the standalone feature of an XGBoost model, achieved by directly prompting ChatGPT to determine if it generated a given text [1]. While this level of accuracy may not be sufficient for a standalone classifier, it suggests that zero-shot classification could be a valuable component within a broader feature-based approach, potentially enhancing overall classification performance and providing meaningful insights when combined with other features.

Consequently, we implement a zero-shot classification approach using Meta's *Llama-3-8B-Instruct* model [23], which was possible with ITU's HPC. This serves as a conceptual implementation of using an LLM API, as using ChatGPT or an API is infeasible for this amount of data, and hosting an LLM as a feature for this task is unrealistic in a real-world setting. The process involves defining system and user roles to structure the interaction with the model. The system role provides a prompt that instructs the AI to respond with 'yes' or 'no' depending on whether the text is AI-generated, while the user role supplies the text to be classified. The model then determines whether the text is AI-generated by responding with the appropriate answer.

Initial experiments revealed significant variations in results based on slight changes in the prompt, necessitating a structured approach to prompt selection. Optimizing prompts for this task involves considering several key factors:

- **Accuracy:** The model should achieve sufficient accuracy to ensure it does not negatively impact the overall feature approach.
- **Label Balance:** The prompt should not be biased toward one label and should maintain a balanced distribution of predictions. We propose a 'label balance score' (LBS) as the minimum ratio between the two labels, multiplied by two. A higher LBS indicates better label balance:

$$\text{LBS} = \min\left(\frac{\text{TP}}{\text{TP}+\text{TN}}, \frac{\text{TN}}{\text{TP}+\text{TN}}\right) \times 2$$

  This metric is important for this task as initial experimentation found accuracies of around 50%, which may imply predictions skewed to one class, given the balanced dataset.
- **None Count:** Since the LLM is instructed to respond with 'yes' or 'no', any other response is recorded as 'None'. An effective prompt should consistently minimize the occurrence of 'None' responses.

To address this optimization task, we define variables within the prompt that we expect to correlate with these factors. We categorize the complexity of these variables into three levels, hypothesizing that more detailed prompts lead to better predictions but may result in more non-responses:

Task Description: By varying the complexity of the task description, we aim to observe changes in accuracy.

- **Simple:** Determine if the following text is AI-generated.
- **Moderate:** Considering typical AI text traits, determine if the following text is AI-generated.
- **Detailed:** Considering typical AI text traits such as repetitive phrasing, overly formal language, and minimal errors, determine if the following text is AI-generated.

Output Instructions: The second part of the prompt specifies the output instruction, namely to output 'yes' or 'no'. By varying the level of detail in defining what 'yes' and 'no' mean, we anticipate differences in none counts and label balance.

- **Simple:** Respond with 'yes' or 'no'.
- **Moderate:** Respond with 'yes' if the text is AI-generated, 'no' otherwise.
- **Detailed:** Respond with 'yes' if the text is AI-generated, 'no' if the text is human-written.

Yes/No Order: We switch the order of 'yes' and 'no' in the output instructions to test the effect on label balance. Emphasizing the 'no' label may lead to more balanced predictions, as the LLM appears to prefer responding with 'yes'.

To evaluate the impact of each variable on the success of the prompt, we employ a mixed-effects linear regression model, which accounts for both fixed and random effects [24]. In our context, fixed effects include variables such as the complexity of the task

description and output instructions, which we hypothesize will influence classification performance. Random effects, such as text ID, capture the variability across different texts that is not controlled by the fixed factors.

This method is particularly appropriate for our study as it allows us to isolate the impact of prompt variables on classification performance while accounting for the differences across individual texts. We create all combinations of prompts and evaluate them on a uniform set of 10,000 texts, recording whether each classification was correct. The mixed-effects linear regression model uses these results to estimate coefficients and other statistics, allowing for the identification of which prompt combinations yield the highest performance:

- **Coefficients (Coef.)**: Show each variable's effect on accuracy; positive values improve it, negative values reduce it.
- **Standard Error (Std.Err.)**: Indicate the precision of coefficients; smaller values mean more precision.
- **z-value (z)**: Measure how far coefficients are from zero; higher values mean more significance.
- **P-value (P>|z|)**: Show the likelihood of the effect being due to chance; values below 0.05 are significant.
- **Confidence Intervals ([0.025, 0.975])**: Indicate the range for the true effect size with 95% confidence; narrower intervals mean more confidence.
- **Random Effects (Group Var)**: Account for variability between groups (texts) not explained by fixed effects, showing group-specific impact on accuracy.

To further validate these results, we also analyze accuracy, label balance score, and the number of failed predictions (none count). Collectively, these methods enable us to identify the optimal prompt structure for this task.

Evasive Text Generation

To evaluate the robustness of each proposed AI text detector, we further evaluate them by generating new data using an LLM that was unseen in the training data. We hypothesize that more complex models will perform better on evasive texts, namely the fine-tuned DistilBERT model and XGBoost extended feature set.

We generate these texts with the Llama-3-8B model by rephrasing 5000 AI texts from the test set, and create a balanced evasive test set by combining these rephrased texts with 5000 human texts from the test set. Furthermore, we structure the evasive prompt generation similarly to the zero-shot LLM prompt optimization, formulating hypotheses about what might be effective and defining variables within our prompts. The task descriptions provide context to the AI, while the output instructions specify expectations for the generated text. The prompts are defined as follows:

Task Description:

- **Control:** "Rewrite the following text in your own style and tone."
  - This prompt acts as a baseline for comparing the results of other evasive texts. With this prompt, we expect the LLM to utilize its inherent training patterns and tendencies, which will differ from the LLM that generated the text. This can be perceived as a form of unique style or tone specific to the model.
- **Basic:** "Rewrite the following text so it won't be detected by AI detection tools."
  - This represents a basic understanding of AI text detectors, relying on the AI to avoid detection by rewriting the text.
- **Advanced:** "Considering typical AI text traits such as repetitive phrasing, overly formal language, and minimal errors, rewrite the following text so it won't be detected by AI detection tools."
  - This assumes an advanced knowledge of AI text detectors, targeting specific characteristics of AI-generated text.

Output Instruction: "Respond with only the rewritten text."

This approach provides three levels of AI text rephrasing, facilitating a comprehensive evaluation of our models' effectiveness. The output instruction is specified based on preliminary experiments, which often revealed undesirable comments or preambles, such as "Here is the rewritten text." These occurrences will be quantified using Regex, and alongside instances where the LLM refuses to rephrase the text, they will serve as metrics to evaluate the prompt's effectiveness, particularly in terms of complexity.

We visualize the features of all texts, including original AI and human texts, to examine the presence of specific features across different rephrased texts. For example, we might expect the advanced rephrased texts to contain more errors on average, given the prompt. The effectiveness of the text obfuscation can be evaluated based on the mean feature values for each text type. This analysis could provide insights into the complexity, grammatical correctness, and repetition within the advanced texts.

Results

Evasive Text Metrics

To assess the effectiveness of our evasive prompts, we collected various metrics from the generated texts, including the number of incomplete generations (where the AI could not rephrase the text), introductory deviations (texts beginning with phrases like "Here is the rewritten text"), and concluding deviations (texts ending with similar unnecessary phrases). These metrics provide insights into how well the LLM adhered to the output instructions based on the complexity of the task descriptions.

Evasive Prompt	Incomplete Generations	Introductory Deviations	Concluding Deviations
Control	6	487	34
Basic	54	4071	159
Advanced	30	4387	160

Table 1: *Statistics of evasive text issues for different prompts.*

These statistics indicate how well the LLM performed with each evasive prompt. The higher number of incomplete generations and deviations in the basic and advanced prompts suggest that more complex instructions confused the LLM, leading to incomplete or improperly formatted responses. This pattern highlights a clear relationship between prompt complexity and the number of issues observed. Despite these challenges, the increased complexity in these prompts may enhance their effectiveness for evasion.

To ensure the quality of our dataset, we removed incomplete generations and used regex to trim introductory and concluding deviations. This cleaning process ensures that the remaining data accurately reflects the intended rephrasing without unwanted elements and minimizes data leakage.

Feature Extraction Results

Table 2 presents the average values of 19 out of 41 features (excluding the 500 TF-IDF and 385 Sentence-BERT features) that yield the most insightful and interpretable results. Although these metrics do not indicate a feature's utility in a model, it provides insights into our hypotheses about AI text patterns and whether these trends extend to new LLMs. The features are sorted according to their relative difference between the 'human' and 'ai' columns, and are not presented as standardized.

The higher counts of quotation marks, discourse markers, and words in AI texts compared to human texts suggest that evasive texts appear more human, emphasizing the limitations of a feature approach that may overfit on training data. Additionally, the *perplexity* of AI texts (shown in table 3) is extremely high, indicating they are generally a poorer match for the model used to compute perplexity, compared to human texts. This contradicts our expectation, as GPT-2, used for perplexity calculations, also generated most of the AI texts, leading us to anticipate low perplexity. It is possible that this difference is in part due to variations in the generation process, such as sampling methods or temperature settings, but we make the same observation for the 'advanced' Llama-3 evasive text, which may be evidence of the perplexity model being outdated or inconsistent. To further investigate these differences, we also calculate the median: These median results align more closely with our expectations, suggesting that the means were significantly skewed by outliers, indicating a non-normal distribution for this feature. We find that the median of *mean_perplexity* is consistently low for AI-generated text, validating its relevance as a feature for this approach and dataset. Additionally, the slightly higher perplexity observed in the basic and advanced evasive texts suggests some success in their ability to mimic human-like text.

The summary statistics also give an idea of the validity of our claims on patterns of AI text. AI texts more frequently contain list items, especially in Llama-3, indicating a characteristic of newer LLMs. Our hypothesis on passive voice use is also confirmed, as it appears more in AI texts, though not

feature	human	ai	control	basic	advanced
quotation_count	1.85	6.85	2.08	2.71	2.70
sent_per_par_std	1.74	0.83	0.49	0.53	0.51
list_item_count	0.22	0.38	0.41	0.57	0.53
passive_voice_count	1.47	2.41	1.37	1.54	1.47
words_per_par_std	31.34	21.41	10.29	12.00	11.46
stop_word_count	128.72	184.00	104.24	102.23	99.17
discourse_count	22.70	32.30	20.63	18.85	19.27
dep_clauses_avg	1.17	1.65	1.53	1.34	1.33
error_count	8.40	6.01	1.93	2.28	2.45
word_count	332.19	443.62	265.51	266.72	270.81
burstiness	3.57	4.67	2.90	3.00	2.93
multi_blank_count	7.41	9.62	3.45	5.14	5.21
uniq_words_per_sent	17.08	21.87	21.01	20.44	20.45
syntactic_depth_avg	5.53	6.95	6.89	6.93	6.86
flesch_reading_ease	73.57	62.77	61.22	49.20	50.17
llm_pred	0.92	0.84	0.75	0.93	0.89
unigram_entropy	6.52	6.40	6.38	6.20	6.31
trigram_entropy	7.94	7.99	7.74	7.50	7.61
bigram_entropy	7.77	7.77	7.58	7.34	7.46

Table 2: *Summary statistics of features for human and AI texts (340k texts each) and various evasive texts (5k texts each).*

statistic	human	ai	control	basic	advanced
mean	513.47	22655.04	44.55	71.78	16580.69
median	156.458	33.896	28.789	36.185	34.533

Table 3: *Mean and median of 'mean_perplexity' across different test sets*

significantly in evasive texts, which is problematic. AI texts are also more complex, with more dependent clauses, deeper syntactic trees, lower readability, and more unique words per sentence. They exhibit more repetition, with higher burstiness and a lower standard deviation in words per paragraph, indicating less uniform sentence lengths. Results for bi- and tri-gram entropy are mixed, suggesting limited usefulness of these features in detecting repetition. Additionally, higher unigram entropy seems to be slightly more expressive of human texts rather than AI texts, which goes against our expectations of it.

Evasive texts do not properly represent a sequence of "difficulty," showing somewhat random results across features. The control set of evasive texts shows the lowest readability and perplexity, contrasting sharply with our expectations, potentially suggesting that prompts like "rephrase in your own style" could be effective evasive strategies. The efficiency of these evasive prompts is partially demonstrated by lower llm_pred values, indicating more 0-predictions, with the control evasive text being the hardest and basic the easiest. There are only marginally more errors (blank count, error count) in advanced texts and marginally less repetition and complexity, despite instructions to rephrase text based on these features.

Overall, while these results confirm some hypotheses about AI text patterns, they also reveal limitations in our feature approach, particularly regarding overfitting on training data. Introducing unseen human texts could better contrast the evasive text feature results, and using more modern LLMs for AI text data and perplexity calculations could enhance the robustness of our models and provide a more accurate representation of current standards. Additionally, the inconsistent metrics of evasive texts call for a more structured approach to evasive prompts.

Zero-shot LLM Optimization

We conducted a comprehensive prompt optimization experiment due to initial findings showing significant variability with different prompts. We tested 18 prompts in a structured experiment:
This suggests prompts perform better when predictions are skewed towards one class, raising questions about why accuracy is worse than random guessing on a balanced dataset. This skewing may derive from subtle biases in prompt phrasing, guiding the LLM toward a particular label and highlighting the sensitivity of the LLM's output to the prompt's wording. Despite being less than 50% accurate, the LLM's predictions may still capture specific text features that, when combined with other features, could enhance overall model performance. Ideally, we want an accurate prompt that maintains balanced predictions,

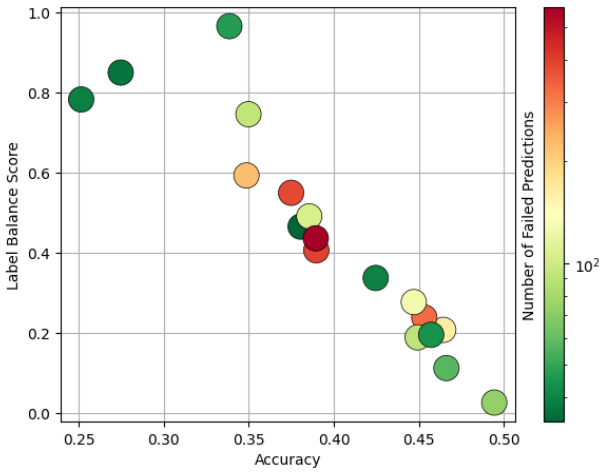

Figure 1: *Scatter plot of accuracy vs. label balance for different prompts, with color indicating the number of failed predictions.*

rather than one that skews towards a single class.

To find the best prompt, we refer to the results of the mixed-effect linear regression model:

Feature	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	0.464	0.004	115.608	0.000	0.456	0.472
task_description[T.Moderate]	-0.062	0.002	-32.430	0.000	-0.066	-0.059
task_description[T.Simple]	-0.060	0.002	-31.389	0.000	-0.064	-0.057
output_instruction[T.Moderate]	0.038	0.002	20.013	0.000	0.035	0.042
output_instruction[T.Simple]	-0.037	0.002	-19.033	0.000	-0.040	-0.033
yes_no_order[T.Yes/No]	-0.067	0.002	-42.798	0.000	-0.070	-0.064
Group Var	0.124	0.006				

Table 4: *Mixed Linear Model Regression Results*

These results indicate that the best prompt uses a detailed task description and moderate output instructions:

- **Task Description Complexity**: Moderate (-0.062) and simple (-0.060) task descriptions decrease accuracy compared to detailed descriptions, confirmed by significant z-values and narrow confidence intervals.
- **Output Instruction Detail**: Moderate output instructions (0.038) improve accuracy, while simple instructions (-0.037) reduce it, as confirmed by significant z-values.
- **Yes/No Order**: Placing 'Yes' before 'No' (-0.067) decreases accuracy. This is highlighted by the high absolute z-value (-42.798), suggesting that the option order significantly influences performance.

Additionally, the random effects variance of 0.124 indicates significant text-specific variability, suggesting the need to account for additional text-related factors not captured by the fixed effects. The P-values of 0 indicate that the results are highly statistically significant, and the small standard errors reflect precise estimates.

From these results, we deduce that detailed task descriptions and moderate output instructions yield higher accuracy. The optimal prompt configuration involves a detailed task description, moderate output instructions, and *No* before *Yes* in responses. However, the prompt experiment results show that the label balance for this prompt is only 2%.

Output Instruction	Accuracy	Precision	Recall	F1 Score	NC	LBS
Detailed	0.464	0.480	0.861	0.616	162	0.207
Moderate	0.494	0.497	0.982	0.660	73	0.026
Simple	0.425	0.454	0.757	0.568	40	0.337

Table 5: *Performance metrics, including the number of failed predictions (NC) and the label balance score (LBS), for classifications of 10,000 texts using prompts with a 'detailed' task description and different output instructions.*

There is a clear trade-off between simplification of output instructions and "None Count" (NC), i.e., the AI failing to output an acceptable response. The "Simple" approach has a higher label balance score (LBS) and a lower NC than the detailed output instruction but a 4% lower accuracy. Given that a more detailed prompt may lead to more thoughtful predictions, we choose the 'Detailed', 'No/Yes', 'Detailed' prompt for use in the zero-shot LLM:

> *"Considering typical AI text traits such as repetitive phrasing, overly formal language, and minimal errors, determine if the following text is AI-generated. Respond with 'no' if the text is human-written, 'yes' if the text is AI-generated."*

Grid Search Results

- **XGBoost (Baseline/Extended)**
  Best Parameters: reg_lambda: 0.01, reg_alpha: 0, n_estimators: 3500, min_child_weight: 5, max_depth: 5, learning_rate: 0.1, colsample_bytree: 0.8
  Runtime: 7.5 hours
- **XGBoost (TF-IDF)**
  Best Parameters: reg_lambda: 0.1, reg_alpha: 0, n_estimators: 4000, min_child_weight: 3, max_depth: 5, learning_rate: 0.1, colsample_bytree: 0.6
  Runtime: 0.5 hours
- **DistilBERT (Fine-tuned)**
  Best Parameters: learning_rate: 4.46e-05, num_train_epochs: 3, per_device_train_batch_size: 64, weight_decay: 0.0041, warmup_steps: 151
  Runtime: 27.5 hours

The identical hyperparameters for the feature-based XGBoost models suggest that the extended feature set does not significantly impact hyperparameter tuning or runtime. The TF-IDF approach had somewhat similar parameter results, with more regularization and the maximum number of estimators, indicating that more estimators may further enhance performance.

The XGBoost models used 64-core CPU parallelization, while DistilBERT employed Tesla V100 GPU acceleration. Although these runtimes are not directly comparable due to varying factors like data/model overhead and different HPC node hardware, they provide a general idea of the resources required. These times exclude additional durations for feature extraction, such as calculating text perplexities and LLM predictions for the hand-crafted feature approaches.

Classification Results

With the LLM and feature models optimized, we can evaluate their performance on our diverse test sets.

Model	Accuracy	Precision	Recall	F1
DistilBERT (Fine-tuned)	98.3%	97.0%	99.7%	98.3%
Llama-3-8B (Zero-shot)	46.4%	47.9%	84.5%	61.2%
XGBoost (Baseline)	96.4%	95.6%	97.3%	96.5%
XGBoost (Extended)	96.5%	95.7%	97.4%	96.6%
XGBoost (TF-IDF)	85.7%	83.0%	89.7%	86.2%

Table 6: *Performance metrics on 34,000 AI + 34,000 human texts*

The results show that the fine-tuned DistilBERT model outperforms all other models across all metrics, emphasizing the superiority of pre-trained language models for this task. Despite being a more generic pre-trained LLM, DistilBERT significantly surpasses our feature-based approaches, highlighting the potential for further improvements with more advanced fine-tuned models.

Among feature-based models, the extended XGBoost model demonstrates a marginal improvement over the baseline, indicating the added value of novel features in detecting AI-generated text. The TF-IDF XGBoost model, although not as strong as the feature-based models, still performs well, considering its simplicity compared to the other models.

The grid search runtime results correlate with these performance metrics, suggesting a trade-off between computational requirements and performance. Feature-based XGBoost models can be executed without a GPU, offering a balance of computational efficiency and accuracy. In contrast, the DistilBERT model, despite its high performance, demands significant computational resources. The Llama-3-8B model, with its low accuracy, serves as a conceptual baseline, illustrating the challenges of zero-shot AI detection. This highlights the importance of training models specifically for the task of AI text detection rather than relying on prompting an AI directly to make a classification.

Overall, the findings suggest that while fine-tuned LLMs like DistilBERT offer superior performance, feature-based approaches remain competitive and provide a feasible alternative for environments with limited computational resources. Further research into advanced fine-tuning techniques and novel features is recommended to bridge the performance gap.

Model	Accuracy	Precision	Recall	F1
DistilBERT (Fine-tuned)	87.8%	95.9%	79.1%	86.7%
Llama-3-8B (Zero-shot)	41.9%	45.1%	75.3%	56.5%
XGBoost (Baseline)	71.5%	63.9%	98.8%	77.6%
XGBoost (Extended)	73.2%	56.4%	98.7%	78.7%
XGBoost (TF-IDF)	72.3%	77.2%	63.2%	69.5%

Table 7: *Performance metrics on 5,000 AI-rephrased + 5,000 human texts*

The results show a decline in metrics across all models when compared to the ordinary test set, indicating that rephrasing texts with a different LLM impacts performance, reflecting a real-world scenario. DistilBERT remains the top-performing model, demonstrating its robustness and ability to generalize well to new data, including text generated by a different LLM.

The extended feature XGBoost model surpasses the baseline in accuracy and F1 score, but is outperformed in recall. This indicates that the baseline model more effectively identifies AI-generated texts, which is significant with this test set, as the AI texts are rephrased by an unseen LLM. This suggests the additional features in the extended model may be overly reliant on patterns from the training data, and not generalize as well to new LLM texts, at least for this specific prompting method, which was to rewrite the text in a different style and tone.

We also see higher recall in the TF-IDF approach than the feature approaches, indicating its ability to generalize well with unseen LLM-generated text. Overall, these results underscore the importance of evaluating models on varied datasets to ensure robustness and generalization.

Model	Accuracy	Precision	Recall	F1
DistilBERT (Fine-tuned)	91.5%	96.2%	86.4%	91.0%
Llama-3-8B (Zero-shot)	50.6%	50.3%	92.8%	65.2%
XGBoost (Baseline)	68.1%	61.4%	97.6%	75.4%
XGBoost (Extended)	69.8%	62.7%	97.8%	76.4%
XGBoost (TF-IDF)	81.4%	81.3%	81.5%	81.4%

Table 8: *Performance metrics on 5,000 basic evasive AI-rephrased + 5,000 human texts*

These results indicate that basic evasive texts are only partially effective at concealing AI origin, as evidenced by the decreased performance of handcrafted feature approaches and improved performance of fine-tuned and TF-IDF models. The LLM likely rephrased texts to minimize AI-indicative features, reducing the efficacy of feature-based models. However, the better performance of other models compared to the 'control' AI-rephrased texts suggests that prompting the LLM to rewrite in a different style and tone might be a more effective evasion strategy, as it more comprehensively alters the text.

Notably, the TF-IDF XGBoost model generalizes well, outperforming feature-based models. This suggests that the simplicity and robustness of the TF-IDF approach effectively capture consistent features across different LLMs, even against evasion techniques.

Interestingly, the zero-shot Llama-3-8B model achieves over 50% accuracy for the first time, an 8.7% increase from the previous test set, potentially due to the ineffectiveness of the prompting method.

Additionally, the extended feature-based approach outperformed the baseline on all metrics in this test set, contradicting the previous findings and indicating that the novel features may offer better generalization with different LLMs.

Model	Accuracy	Precision	Recall	F1
DistilBERT (Fine-tuned)	90.4%	96.1%	84.2%	89.8%
Llama-3-8B (Zero-shot)	48.6%	49.2%	88.8%	63.4%
XGBoost (Baseline)	57.7%	54.2%	98.5%	70.0%
XGBoost (Extended)	60.3%	55.8%	98.5%	71.3%
XGBoost (TF-IDF)	80.0%	80.8%	78.8%	79.8%

Table 9: *Performance metrics on 5,000 advanced evasive AI-rephrased + 5,000 human texts*

The advanced evasive test set results confirm the basic set findings, showing similar patterns but with decreased performance, indicating this is a more effective prompt for AI detection evasion. The feature-based models are particularly impacted, highlighting the prompt's success in rewriting text to avoid complexity and repetition. These results suggest that complexity and repetition are indicative of AI text, and that prompting the LLM to avoid specific features is more effective than relying on the LLM's judgment alone.

These results highlight the potential for future research into prompting strategies that instruct AIs to rewrite text to evade detection. Despite the decreased performance, the accuracies of the TF-IDF, fine-tuned, and zero-shot LLM models remain higher than those for the control set, which was unexpectedly an effective prompt for obfuscating AI text origins.

SHAP Analysis of Model Predictions

In our evaluation of feature models, we employ SHAP (SHapley Additive exPlanations) to interpret the impact of each feature on the model's predictions. The SHAP values on the x-axis represent the impact of each feature on the prediction, with higher values indicating a more significant contribution to the positive class (AI-generated text) and lower values indicating a contribution to the negative class (human-written text). Features on the y-axis are sorted by importance, with the most influential features at the top.

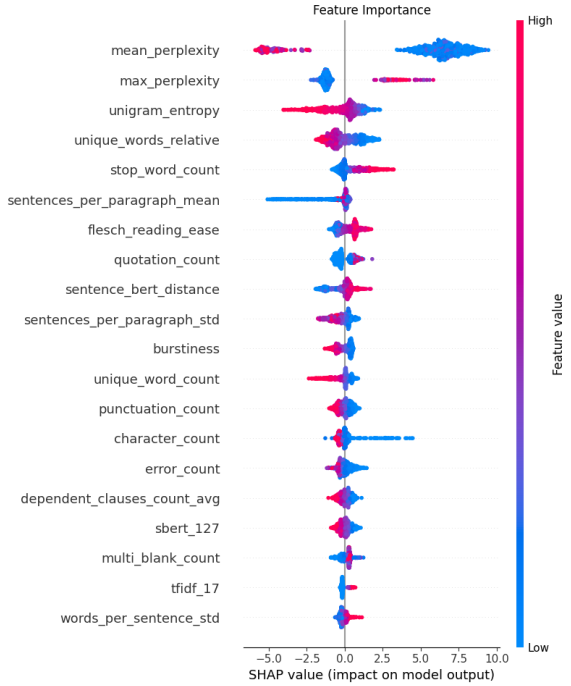

Figure 2: *SHAP summary plot for the extended feature model*

The SHAP summary plot in Figure 2 reveals the most important features in determining text as AI-generated:

- **Mean Perplexity and Max Perplexity:** These features are among the top in terms of importance. Low mean perplexity is the most important for classifying text as AI-generated, while high values classify text as human-written, indicating that human texts are less predictable to GPT-2, which is consistent with our feature extraction results. Conversely, max perplexity has an opposite correlation, with medium to high values indicating AI-generated text, and low values somewhat important for human texts. This discrepancy could be due to excessive outliers, potentially created using sampling methods or temperature settings.
- **Novel Features:** Three novel features appear in the top 20, a significant achievement given the extensive feature set (41 hand-crafted features, 500 TF-IDF features, and 385 Sentence-BERT features). These features include:
  - **Unigram Entropy (3rd best feature):** High values of unigram entropy indicate human-written texts, contrary to our hypothesis that AI texts are more complex.
  - **Burstiness (11th best feature):** Higher values for burstiness indicate human-written texts, contrary to our expectations regarding repetition.
  - **Average Number of Dependent Clauses (16th best feature):** Higher values indicate human text, suggesting that human texts are more complex according to these metrics.

These novel features perform comparably in identifying both human and AI text. While our initial assumptions about complexity and repetition may not hold, these features remain effective in distinguishing between human and AI texts. Conversely, AI texts are harder to read (Flesch Reading Ease), rely more on stop words, and have a lower proportion of unique words compared to human texts. These features, which seemingly contradict each other in terms of complexity, suggest that both human and AI texts exhibit complexity, albeit in different ways.

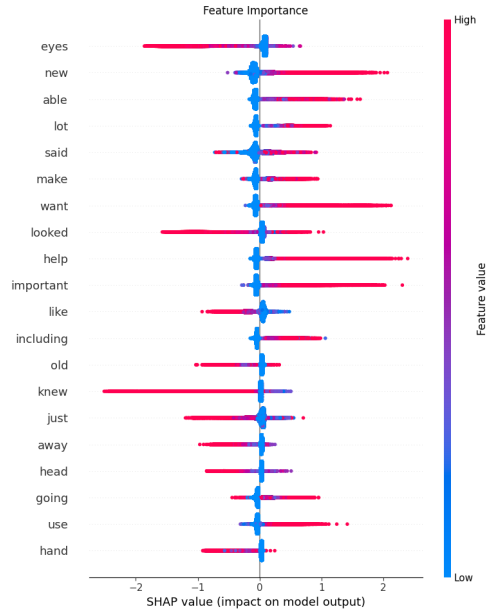

Figure 3: *SHAP summary plot for the TF-IDF model*

The SHAP summary plot for the TF-IDF model demonstrates the importance of specific terms in determining the AI-generated nature of texts. Key terms and their contributions to the model's predictions include:

- **Instructional or Request Words:** Words like "use," "help," "want," and "able" are more frequent in AI-generated texts, reflecting the directive nature of AI text generation.
- **Past vs. Present Tense:** Human texts favor past tense words like "looked" and "knew," while AI texts may use present tense words like "use" and "want" more frequently.
- **Sentiment Words:** Words like "like" appear more in human texts, potentially due to its use as slang or to convey sentiment.
- **Temporal Words:** "Old" appears more frequently in human text, while "new" is more common in AI text.

- **Physiological Human Features:** Terms like "eyes," "hand," and "head" are more frequent in human texts, which might be due to dataset biases or a different conceptual focus between human and AI texts.

This method of analyzing terms through TF-IDF could aid in detecting AI text by manual inspection, as suggested by studies like the "delve" study [11]. Despite not identifying highly descriptive words such as "delve," future research could investigate TF-IDF features with more expressive datasets, such as texts from newer models like GPT-4 and high-level human texts like research articles. This approach could help identify distinct terminology that makes AI text detection easier by eye.

These SHAP analyses provide valuable insights into the distinguishing features of AI-generated text and demonstrate the efficacy of both feature-based and TF-IDF models in AI text detection. The interpretation of SHAP values emphasizes the critical role of specific features and terms, enhancing our understanding of the underlying patterns in AI-generated content.

Discussion

Academic Applications of AI Text Detection

The emergence of AI text detectors presents both opportunities and ethical dilemmas within academic integrity. While these tools are invaluable in identifying instances of AI-assisted writing, their application in an academic context requires careful consideration. Research papers [3] and the tools themselves warn academic authorities to use the detectors carefully, as they are not concrete evidence of academic misconduct, and can only strongly imply that the text reads as AI written. Nonetheless, there have been instances of expulsion due to students using AI in assignments.

Turnitin recommends strategies[3] to restrict the use of generative AI in academia, such as by ensuring the "prompt" of an assignment is not answerable with AI to an acceptable degree, and for assignments to include personal experience. It is clear that the challenge of limiting generative AI in academia cannot be solved solely with the use of AI text detectors, and it is strongly implied in the research paper [3] that additional preventative measures should be implemented. Some further ideas for preventative measures could be:

[3] https://marketing-tii-statamic-assets-us-west-2.s3-us-west-2.amazonaws.com/marketing/tli_ai_respondinginyourclassroom_guide_uk_0123.pdf

Edit History Analysis This system would enforce an edit history for each submission, for instance via Google Docs. The submissions could be scanned for AI text, and if any is detected, the edit history could be consulted to figure out whether the text was copied and pasted. Potentially, this process could be automated, analyzing the edit history for suspicious typing patterns, outputting a metric of certainty. This could be combined with AI text detectors to collect further proof of academic misconduct.

On the other hand, this preventative measure is not bulletproof. An irrepressible adversary will always have the upper hand in the realm of AI text detection, as seen with evasive prompts. For instance, to fake an edit history as not being copy-pasted, the adversary could employ a Python script that utilizes 'pyautogui' to read from a document. With this, they could mimic human typing with random intervals between keystrokes and sentences, effectively bypassing detection systems that monitor typing patterns or edit histories.

Reference Verification One pitfall of modern LLM's is their inability to cite their own information accurately. LLM's, being machine learning models, learn from their training data enough to be able to create responses on unseen data. This makes their inner workings something of a black box, especially with non-open-source models such as ChatGPT. As they are trained on such diverse and extensive data, it can be hard to determine where they attained certain information from, and whether it is fabricated. When prompted to cite all information the LLM supplies, it may reference incorrect or nonexistent articles and links.

By mandating that students cite their assignments extensively, they could be deterred from using LLM's in their assignments. Additionally, algorithms could be implemented to determine whether the given citation actually describes the corresponding information, and whether the citation even exists.

Employing these preventative measures in combination with a robust AI text detector capable of detecting evasive prompt-tuned text could further deter students from using AI in their coursework. Utilizing more than just AI text detectors provides evidence closer to concrete proof of academic misconduct, leaving it up to the academic institution whether to use this evidence as punishable judgement. Research could be conducted to see how much more AI text is detected before and after the preventative measures are implemented, given the students are aware of them. This marks an ethical use of AI text detection, as they are not being used as concrete evidence, but rather used for guiding decisions within preventative

measures.

AI Text Detection in Other Fields

The main concern with using AI text detectors in academic contexts is their inability to collect irrefutable proof of misconduct, as the action of expelling or punishing a person without complete evidence is very serious and potentially unethical. However, this does not extend to other fields, such as a company using AI text detection to monitor their own applications. AI text detection could have a prominent role in content moderation, in the effort of purging databases of fake content, such as social media posts and articles. This has less extreme and unethical consequences, as it is isolated to the company's own application, not necessarily an individual's reputation.

For example, Google could implement AI text detection in their PageRank algorithm, which ranks search results based on how many times the link to said page is referenced to on other pages. There are likely many pages exploiting AI text generation for this purpose, creating fake websites whose sole purpose is to refer to a certain link, increasing its visibility on Google's search engine. By implementing AI text detection into their PageRank algorithm, they could exempt AI-created web pages and articles from being considered in their rankings. This idea of applying AI text detection to content moderation could be extended to many other applications.

Limitations and Future Work

While this study provides valuable insights into AI text detection, several limitations should be acknowledged and addressed in future research. The dataset employed, though comprehensive, may not fully capture the diversity and quality of AI-generated content, as it predominantly comprises texts from specific large language models (LLMs) such as GPT-2 and GPT-3. To enhance the generalizability and robustness of detection models, it is necessary to expand the dataset to include texts generated by newer and more diverse AI models, including GPT-4 and other state-of-the-art LLMs.

Additionally, this study primarily focuses on document-level detection methods, which, while effective, may not capture the finer granularity required for certain applications. Future work should explore sentence-level detection techniques to provide more precise and nuanced identification of AI-generated text within longer documents.

Finally, the evasive text generation techniques utilized in this study were relatively basic. Future research should include more sophisticated evasive text generation techniques to better test the robustness of detection models.

Conclusion

The differentiation between AI-generated and human-written texts is increasingly critical in maintaining academic integrity, ensuring content authenticity, and supporting ethical AI use. This thesis presented a comprehensive feature-based classification approach that combines established NLP features with novel attributes tailored to capture the unique characteristics of AI-generated text.

Our results demonstrate that while the established features remain effective, the inclusion of new features enhances detection accuracy, even in the presence of evasive texts. Specifically, the fine-tuned DistilBERT model emerged as the best performer, achieving the highest accuracy and robustness across various test sets, including those with evasive texts. The TF-IDF-based XGBoost model also showed good generalization capabilities, particularly against unseen LLM-generated text.

The SHAP analysis provided valuable insights into the most influential features for AI text detection. Key features included mean perplexity, unigram entropy, and burstiness, highlighting the different ways AI and human texts exhibit complexity and repetition. These findings underscore the importance of continuous feature engineering and the development of advanced LLM models to keep pace with advancements in AI text generation.

Our study highlights significant challenges, such as the potential for overfitting and the limitations of current models when faced with sophisticated AI-generated texts designed to evade detection. These findings underscore the need for continuous improvement in feature engineering and model robustness to keep pace with advancements in AI text generation.

Our contribution to the field of AI text detection includes providing a comprehensive comparative analysis of the feature-based approach, investigating features that improve the model's performance. While our results indicate that large language models (LLMs) generally offer superior detection capabilities, the feature-based approach also proves to be robust and effective. By establishing both upper and lower bounds, we contextualized the performance of the feature-based model and highlighted its practical utility.

Additionally, we define a comprehensive methodology for assessing the effect of prompt variation on output quality, particularly through text paraphrasing and zero-shot text detectors. Our investigation reveals effective prompts for both detecting

and avoiding AI text detection, providing valuable insights into the dynamics of AI-generated text.

In conclusion, while fine-tuned LLMs offer superior performance in AI text detection, feature-based approaches remain valuable and practical, especially in scenarios with limited computational resources. The study's insights into evasive text generation and prompt optimization further contribute to the development of more robust and effective AI text detection methods. Future research should focus on addressing the limitations identified and exploring new strategies to enhance the robustness and applicability of AI text detection techniques.

References

[1] Lorenz Mindner, Tim Schlippe, and Kristina Schaaff. "Classification of Human- and AI-Generated Texts: Investigating Features for ChatGPT". In: *Lecture Notes on Data Engineering and Communications Technologies*. Springer Nature Singapore, 2023, pp. 152–170. ISBN: 9789819979479. DOI: 10.1007/978-981-99-7947-9_12. URL: http://dx.doi.org/10.1007/978-981-99-7947-9_12.

[2] Rexhep Shijaku and Ercan Canhasi. "ChatGPT Generated Text Detection". In: *ResearchGate* (Jan. 2023). DOI: 10.13140/RG.2.2.21317.52960.

[3] Debora Weber-Wulff et al. "Testing of detection tools for AI-generated text". In: *International Journal for Educational Integrity* 19.1 (Dec. 2023). ISSN: 1833-2595. DOI: 10.1007/s40979-023-00146-z. URL: http://dx.doi.org/10.1007/s40979-023-00146-z.

[4] Tharindu Kumarage et al. *How Reliable Are AI-Generated-Text Detectors? An Assessment Framework Using Evasive Soft Prompts*. 2023. arXiv: 2310.05095 [cs.CL].

[5] Pengyu Wang et al. "SeqXGPT: Sentence-Level AI-Generated Text Detection". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1144–1156. URL: https://aclanthology.org/2023.emnlp-main.73.

[6] Harika Abburi et al. *A Simple yet Efficient Ensemble Approach for AI-generated Text Detection*. 2023. arXiv: 2311.03084 [cs.CL].

[7] artem9k. *AI Text Detection Pile*. https://huggingface.co/datasets/artem9k/ai-text-detection-pile. MIT License, Accessed: 2024-05-19. 2023.

[8] OpenAI. *GPT-2 Output Detector Demo*. https://openai.com/research/gpt-2-output-dataset. Accessed: 2024-05-19. 2023.

[9] University of Nevada, Las Vegas. *Use of Generative AI in Research: Perplexity and Burstiness*. Accessed: 2024-03-14. Mar. 2024. URL: https://guides.library.unlv.edu/c.php?g=1361336&p=10054021.

[10] Irene Solaiman et al. *Release Strategies and the Social Impacts of Language Models*. 2019. arXiv: 1908.09203 [cs.CL].

[11] Jim the AI Whisperer. *Why "delve" is the most obvious sign of AI writing*. The Generator, ~10 min read, Updated: May 20, 2024. May 2024. URL: https://medium.com/the-generator/why-delve-is-the-most-obvious-sign-of-ai-writing-fc4c72e74499.

[12] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[13] Vadim Borisov et al. "Deep Neural Networks and Tabular Data: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Dec. 2022), pp. 1–21. DOI: 10.1109/TNNLS.2022.3229161.

[14] Ravid Shwartz-Ziv and Amitai Armon. "Tabular data: Deep learning is not all you need". In: *Information Fusion* 81 (Nov. 2021). DOI: 10.1016/j.inffus.2021.11.011.

[15] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[16] Gareth James et al. *An Introduction to Statistical Learning*. 1st. PDF. New York, NY: Springer, 2013.

[17] Hugging Face. *Text Classification Guide*. https://huggingface.co/docs/transformers/tasks/text_classification. Accessed: 2024-05-30.

[18] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (2019).

[19] Rafael Barbon and Ademar Akabane. "Towards Transfer Learning Techniques—BERT, DistilBERT, BERTimbau, and DistilBERTimbau for Automatic Text Classification from Different Languages: A Case Study". In: *Sensors* 22 (Oct. 2022), p. 8184. DOI: 10.3390/s22218184.

[20] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[21] Jules King et al. *LLM - Detect AI Generated Text*. 2023. URL: https://kaggle.com/competitions/llm-detect-ai-generated-text.

[22] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG].

[23] Meta AI. *Meta-Llama-3-8B-Instruct*. https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct. Accessed: 2024-05-19. 2024.

[24] XA Harrison et al. "A brief introduction to mixed effects modelling and multi-model inference in ecology". In: *PeerJ* 6 (2018), e4794. DOI: 10.7717/peerj.4794.