Detection of AI-generated Text through Supervised Learning

Kevin Li
xl3556@nyu.edu

Jeff Ma
jeff.ma@nyu.edu

Harry Chen
hc2837@nyu.edu

Demi Guo
kg2412@nyu.edu

May 22, 2023

Abstract

We present a technique to distinguish between human-generated and machine-generated text. Our method involves the extraction of five statistical features, including the average sentence length, the ratio of repetitive words, entropy, punctuation, and digit numbers. These features are combined with TF-IDF features. Then, we optimize hyperparameters using Grid Search and employ machine learning models, including logistic regression (LR), support vector machine (SVM), and random forest (RF) to classify AI-generated excerpts. Our approach is evaluated on 6 datasets, each containing 5000 samples of human and 5000 samples of machine-generated text, which is divided into training, validation, and testing subsets. By using 3 evaluation metrics, precision, recall, and F-measure, we demonstrate the effectiveness and reliability of our method in accurately distinguishing between human-produced and machine-produced text. We then conduct an error analysis to see the 30 most common misclassified part-of-speech tags. We build a prototype web-based detector application with Flask that can be easily scaled with larger custom data sets.

1 Introduction

As Artificial Intelligence (AI) systems continue to advance, they have become increasingly precise and complex. They are now frequently used in numerous applications, including chatbots and language translation, to generate text (Schmaltz (2018)). While this capability has led to many useful applications, it has also raised concerns about the authenticity of text in various contexts.

Recent advancements in Natural Language Processing have led to diminishing differences between human-generated text and machine-generated text (Clark et al. (2021)). Despite this progress, there remains a need to differentiate text generated by humans from that generated by AI systems. Consequently, our research aims to explore statistical techniques for making this distinction.

In order to differentiate between text created by humans and that generated by machines, we will extract a variety of statistical features (Solaiman et al. (2019)). These features will include measures such as average sentence length, the frequency of repeated words, text entropy, punctuation count, and number count. We will then combine these extracted statistical features with TF-IDF features for our analysis (Gehrmann et al. (2019)).

We employ supervised learning algorithms in our project to build a classification model. Specifically, we will use algorithms such as logistic regression, support vector machine (SVM), and random forests. By training our model on labeled datasets of both human-written text and machine-generated text, we aim to create a reliable classifier that can effectively differentiate between the two. We will also evaluate the performance of each algorithm and conduct an error analysis.

2 Related Work

Statistical features have been applied in detectors for AI-generated text. Solaiman et al. (2019) use tf-idf vector on top of a logistic regression model to distinguish WebText articles (online webpages) from text generated using GPT-2 models with considerably high accuracies (ranging from 88% to 74% based on the parameters in GPT2 model used). Our work first reproduces similar results. Other than tf-idf vectors, Gehrmann et al. (2019) show text entropy can be used to discriminate machine-generated text and Ippolito et al. (2020) show that text detection systems are sensitive to excerpt length. Li et al. (2019) combined BERT and logistic regression for propaganda detection, with L-BFGS solver, L2 penalty, and a regularization strength of 1.0. The second model employed in our study is the Support Vector Machine (SVM), which is inspired by the work of Bation et al. (2017). They developed an SVM document classifier for news classification and demonstrated that SVM classifiers are capable of handling large feature spaces and performing effectively on text data classification tasks. We also employ Random Forests in our study, inspired by the work of Xu and Jelinek (2004). The advantages outlined in their paper (useful for POS tagging, parsing, named entity recognition) provide insights into the potential reasons behind the performance of the Random Forest model in our project.

3 Features

3.1 Statistical Features

3.1.1 Average Sentence Length

Average Sentence Length is the average number of words in a sentence in a given excerpt. Together with the features of the words in an excerpt, it provides important information about the complexity and structure of the text being analyzed. It has been observed that machine-generated text typically has shorter sentence lengths compared to human-generated text. Mathematical Formula:

- num_of_sent: total number of sentences in the given text
- total_words: total number of words in a given text
- avg_sentence: average sentence length in a given excerpt

avg_sentence = num_of_sent/total_words

The Natural Language Toolkit (NLTK) package is used to tokenize the text into sentences and words. NLTK provides $sent\_tokenize$ for separating a given excerpt into sentences using recommended tokenizer and $word\_tokenize$ separating a sentence into word tokens. An excerpt can be passed in our `sentence_length` function, and the average sentence length feature defined above will be returned and then used for our classifier.

3.1.2 Repeating Word Ratio

Machine-generated texts have more repetitions than content that has been written by a human. The repetitive word ratio can be formulated as follows:

$$r = \frac{n}{N}$$

- r: Repetitive word ratio
- n: Number of repeated words
- N: Total number of words

The `repetitivewords` function first tokenizes the text into words before utilizing the NLTK library to determine each word's synonyms. After that, it counts the occurrences of words with related synonyms in the text, giving the numerator for the calculation above. In this study, we try to distinguish between genuine human-authored content and that produced by automated systems using the property of repeating words.

3.1.3 Entropy

Entropy can show the complexity and coherence of a text. The "Entropy" function uses NLTK to tokenize the text into words and calculate the entropy for each text. The entropy (H) of a text with N distinct tokens and a probability of p_i for each token is given by the formula:

$$H = -\sum_{i=1}^{N} (p_i \cdot \log_2 p_i)$$

When $H = 0$, the text is entirely predictable, i.e., if it only contains one token. When $H = \log_2(N)$, the text occurs equally frequently across all N different tokens. Machine-generated text typically has a lower entropy value due to its repeated patterns and lack of semantic coherence, whereas human-authored material often has a higher entropy value due to its greater linguistic complexity and unpredictability.

3.1.4 Punctuation

Another statistical characteristic is the frequency of punctuation. Machine-generated text is based on pre-established algorithms which makes the punctuation more predictable. However, human-generated text could have more variation in the way that punctuation is used as people use punctuations in different ways to convey various meaning. The function "count_punctuation" calculates the average number of punctuation per sentence in a given text. It takes a string of text as input, tokenizes the text into sentences using the nltk library, and counts the total number of punctuation marks in the text using the string library. The formula for this calculation can be expressed as:

- average punctuation per sentence = (total punctuation marks / number of sentences)

3.1.5 Digit Numbers

Machine-generated text is tend to include structured data because of predetermined rules. In contrast, human writers use a wider range of numerical digits to achieve various writing purpose. The function “count_numbers" tokenizes the text into sentences use NLTK libarary. To count the number of numerical digits, the text is then divided into words, and a lambda function combined with the filter function is used to detect words that contain digit. Here is the formula to calculate the average numerical digits:

- average numerical digits per sentence = (total numerical digits / number of sentences)

3.2 Text Representation

3.2.1 TFIDF

The numerical metric, TF-IDF, measures a term’s importance in a document.

TF-IDF score of a term t in a document d is calculated as follows:

- N: Number of term t appears in document d
- T: Total number of terms in document d
- Ttext: Total number of documents
- Tterm: Number of documents with term t in it

$$\text{TF}(t, d) = \frac{N}{T}$$

$$\text{IDF}(t) = \log_e \left(\frac{T}{Tterm} \right)$$

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

By combining the term frequency in the document and its rarity in the corpus as a whole, with other statistical features, including sentence length, repetitive words, and entropy, and using them to train a machine learning algorithm, the model can distinguish between genuine content written by humans and that produced by automated systems.

4 Data

In machine learning, the data quality is crucial for achieving high precision. To this end, our study utilizes three distinct categories of data: human-written text data, GPT2-generated text data, and GPT3.5-generated data. Each data category requires specific data collection and munging procedures tailored to their unique characteristics. By employing appropriate data preparation techniques, we aim to ensure the high quality and reliability of the data, thereby facilitating the accurate detection of AI-generated text through supervised learning. We prepare 6 data sets with 10000 entries each for our training experiments. There are 5000 entries of human-written excerpts and 5000 entries of machine-produced excerpts. The human-written excerpts are identical among the 6 data sets, and the machine-produced excerpts differ in the models used or generated approaches.

4.1 Human-Written Text Data

We obtained our text data from the New York Times articles published after 2020, beyond the cutoff date used for training the GPT2 model. GLTR from Gehrmann et al. (2019) does not have statistical features for words like COVID-19 or Internet memes. Our work tries to mitigate this problem by applying up-to-date data. Given the enormous volume of New York Times articles available, we employ a random sampling strategy to select a representative subset for our analysis. We utilize `playright` (a JavaScript framework) first to retrieve the links for all the articles from New York Times. For randomly selected links, we extract the text content from articles and also record the article title for future use. Ippolito et al. (2020) show that detection accuracy grows with the length of the text. For our purpose, we drop excerpts with less than `MIN_SENT_LENGTH` (set to 10) tokens.

4.2 GPT2-Generated Text Data

4.2.1 Existing Data

We use the results from Solaiman et al. (2019) as our baseline. Therefore, we built 4 data sets from their published data. The published data contain training, testing, and validation data for 4 different GPT2 models based on the number of parameters – GPT2-Small-117M, GPT 2-medium-345M, GPT 2-large-762M, and GPT2-xl-1542M. We sample 5000 entries randomly from each of them as part of our datasets for comparison.

4.2.2 NYT-like GPT2-Generated Text Data

To ensure the validity of the generated data, we aim to make it as similar as possible to human-written text data. Given that GPT-2 does not have a direct question-answering API like ChatGPT, we employ a specific technique to generate text that closely matches the input. We use a portion of the NYT data as input for the text-generation model, segmenting each paragraph into three equal parts and using the first part as input. We also restrict the output's length to no longer than the original paragraph. In the end, we filter out the NYT filling data to keep the text generated only by GPT2.

Algorithm 1 Generating GPT2 Texts from NYT Paragraphs

```
1: Initialize an empty list gpt2_texts
2: for each paragraph p in NYT_paragraphs do
3:     prompt ← p[0: int(len(p)/3)]
4:     res ← generator(prompt,
5:         max_length=len(p),
6:         num_return_sequences=1)
7:     text ← res[0]['generated_text'].strip()
8:     gpt2_texts.append(text)
9: end for
```

4.3 GPT3.5-Generated Text Data

GPT-3.5 has a question-answering implementation, so we use it to generate our data directly. For each article selected from the NYT, we request GPT-3.5 to produce an article with the same title in the tone of the NYT and then collect the response. Sometimes the title is repeated in the response so we filter that out of our data. This approach ensures that the generated text is coherent, adheres to the style of the source, and maintains relevance to the chosen topic.

Algorithm 2 Generating Articles with GPT-3.5

```
1: for title in titles do
2:     prompt ← Write an article with title 'title'
3:     res ← openai.ChatCompletion.create(
4:             engine = "gpt-3.5-turbo",
5:             prompt = prompt,
6:             max_tokens = 5000,
7:             n = 1,
8:             stop = None,
9:             temperature = 0.7, )
10:    texts.append(res.choices[0].text.strip())
11: end for
```

5 Methodology

In this paper, three supervised learning algorithms are employed for classifying AI-generated text: Logistic Regression, Support Vector Machines (SVM), and Random Forests. These algorithms were selected and fine-tuned based on the optimal parameters obtained through a comprehensive grid search. Below is an in-depth description of each method, including their mathematical foundations and their application in this research.

5.1 Logistic Regression

The logistic regression model is a binary classification algorithm that predicts the probability of an instance belonging to a certain class based on a linear combination of its features. The probability is given by the logistic function:

$$P(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Where $\mathbf{x}$ is the feature vector, $\mathbf{w}$ is the weight vector, and y is the binary class label. The objective function for logistic regression is the negative log-likelihood of the data, which can be written as:

$$J(\mathbf{w}) = -\sum_{i=1}^{N}[y_i \log P(y_i = 1|\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log P(y_i = 0|\mathbf{x}_i; \mathbf{w})]$$

To prevent overfitting, we introduce an L2 penalty term, which is proportional to the square of the L2 norm of the weight vector:

$$L2_{penalty} = \frac{1}{2}\lambda\|\mathbf{w}\|^2$$

The objective function with L2 regularization becomes:

$$J(\mathbf{w}) = -\sum_{i=1}^{N}[y_i \log P(y_i = 1|\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log P(y_i = 0|\mathbf{x}_i; \mathbf{w})] + \frac{1}{2}\lambda\|\mathbf{w}\|^2$$

The regularization parameter λ controls the balance between fitting the data and penalizing large weights. In this paper, we use the parameter C, which is the inverse of λ, namely $C = \frac{1}{\lambda} = 10$.

We use the Newton-CG (Newton-Conjugate Gradient) solver as the optimization algorithm to find the optimal model parameters (weights) that minimize the objective function of logistic regression with L2 regularization.

5.2 Support Vector Machine

Another supervised algorithm used is Support Vector Machine (SVM), a powerful classification algorithm that finds the optimal decision boundary or hyperplane separating data points of different classes, maximizing the margin between the support vectors, which are the data points closest to the decision boundary, and the hyperplane. The hyperplane is defined by:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

SVM aims to minimize the following objective function with a regularization parameter C that balances maximizing the margin and minimizing classification error:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

In this paper, we use two kernel functions for the SVM: linear kernel and Radial basis function (RBF) kernel. The linear kernel is simply the inner product of the feature vectors, while the RBF kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

Where γ is a positive constant controlling the decision boundary's shape.

The SVM optimization problem is solved using the dual Lagrangian formulation, which involves the dual problem, subject to specific constraints. The optimal weight vector $\mathbf{w}$ and bias term b can be obtained from the dual solution.

5.3 Random Forests

Then, we proceed to Random Forest, an ensemble learning method that constructs a multitude of decision trees at training time and outputs the mode of the classes or the mean prediction of the individual trees for classification and regression tasks, respectively. The method combines the power of several weak learners (decision trees) to produce a more accurate and robust prediction by reducing both bias and variance.

The decision trees in a Random Forest are grown by recursively splitting the data into subsets based on a feature that maximizes the information gain, with each split determined by optimizing an impurity criterion, such as the Gini index or entropy. A key aspect of Random Forest is the introduction of randomness during tree construction. At each split, only a random subset of features is considered, and each tree is built using a bootstrapped sample of the original dataset.

5.4 Model in this Paper

In this research, Logistic Regression, SVM, and Random Forests are employed to classify text as either human-generated or AI-generated. The text data is first preprocessed and transformed into a high-dimensional feature space using the Term Frequency-Inverse Document Frequency (TF-IDF) approach. This representation encodes the textual information in a manner suitable for the SVM algorithm, taking into account the importance of words in the text corpus while considering their frequency.

The extracted linguistic features, such as average sentence length, repetitiveness, text entropy, frequency of punctuation, and digit number are combined with the TF-IDF features, providing the supervised learning models with additional context to distinguish between human and AI-generated text. The model is then trained on this combined feature set to learn the underlying patterns and relationships between the features and the target variable.

After performing a Grid Search to identify the optimal hyperparameters, the final Logistic Regression model is constructed with an L2 penalty term and a regularization parameter of C=10, employing the Newton-CG solver for optimization. The final SVM model is built using a linear kernel with a regularization parameter $C = 10$. The final Random Forest model is constructed using 100 trees with a maximum depth of 30. This configuration ensures an appropriate balance between model complexity and generalization capabilities.

6 Experiment

In the experiments section, we focus on the process of selecting the optimal hyperparameters for the machine learning models, specifically using Grid Search for parameter tuning. The main objective is to identify the best combination of hyperparameters for each model, including Logistic Regression, Support Vector Machine, and Random Forest. This is accomplished by assessing the performance of each model on the validation set for various parameter combinations.

Grid Search is an exhaustive search method that evaluates all possible combinations of the specified hyperparameters in a pre-defined search space. In this research, the hyperparameters under consideration for each model are as follows:

Logistic Regression:

- Regularization parameter (C): [0.1, 1, 10]
- Solver: ['newton-cg', 'liblinear']

Support Vector Machine:

- Regularization parameter (C): [1, 10]
- Kernel: ['linear', 'rbf']

Random Forests:

- Number of estimators: [50, 100]
- Maximum depth: [None, 30]

By fitting each model on the validation set using 5-fold cross-validation, Grid Search provides the best combination of hyperparameters for each model, ensuring the optimal performance on unseen data.

7 Results

In this paper, the goal is to accurately classify text samples as either generated by a language model (GPT) or written by a human. To assess the performance of different machine learning models, namely Random Forest, Logistic Regression, and Support Vector Machines (SVM), we use Precision, Recall, and F-measure as evaluation metrics. By comparing these evaluation metrics, we can determine which model performs best on this classification task and select the most suitable one for our application. The evaluation metric has the following mathematical formula: We have divided the results into two classes, the positive class, and the negative class.

- **True Positive (TP):** A text generated by GPT-2 is correctly classified as a GPT-2 generated text by the classifier. This means the classifier correctly identifies it as an artificially generated text.
- **False Negative (FN):** A text generated by GPT-2 is incorrectly classified as a human-written text by the classifier. This means the classifier fails to identify the text as generated by GPT-2 and mistakenly labels it as human-written.
- **False Positive (FP):** A human-written text is incorrectly classified as a GPT-2 generated text by the classifier. This means the classifier mistakenly labels the human-written text as an artificially generated text.
- **True Negative (TN):** A human-written text is correctly classified as a human-written text by the classifier. This means the classifier correctly identifies the text as written by a human and not generated by GPT-2.
- **Precision (positive)** = $\frac{\text{TP}}{(\text{TP}+\text{FP})}$. This metric represents the proportion of correctly identified computer-generated texts out of all texts classified as computer-generated.
- **Recall (positive)** = $\frac{\text{TP}}{(\text{TP}+\text{FN})}$. This metric represents the proportion of correctly identified computer-generated texts out of all actual computer-generated texts.
- **Precision (negative)** = $\frac{\text{TN}}{(\text{TN}+\text{FN})}$. This metric represents the proportion of correctly identified human-generated texts out of all texts classified as human-generated.
- **Recall (negative)** = $\frac{\text{TN}}{(\text{TN}+\text{FP})}$. This metric represents the proportion of correctly identified human-generated texts out of all actual human-generated texts.
- **F-measure for both classes** $= 2 \times \frac{\text{Precision}\times\text{Recall}}{(\text{Precision}+\text{Recall})}$, which is the harmonic mean of precision and recall and provides a balanced measure of the classifier's performance.

Table 2: Results Analysis

Experiment	Average F-measure
Positive	
GPT 2-small-117M	0.785
GPT 2-medium-345M	0.755
GPT 2-large-762M	0.762
GPT 2-xl-1542M	0.775
GPT 2-xl-generated	0.757
GPT 3.5	0.885
Negative	
GPT 2-small-117M	0.776
GPT 2-medium-345M	0.760
GPT 2-large-762M	0.745
GPT 2-xl-1542M	0.774
GPT 2-xl-generated	0.752
GPT 3.5	0.885

8 Discussion

For each model, we take the average of F-measures (Table 2) on all three models and cross-compare

Table 1: Merged Results for GPT Models

Model		Positive Precision	Positive Recall	Positive F-measure	Negative Precision	Negative Recall	Negative F-measure
GPT 2-Small-117M	**RF**	0.727	0.763	0.744	0.750	0.713	0.731
	LR	0.789	0.805	0.806	0.800	0.785	0.793
	SVM	0.805	0.807	0.806	0.806	0.804	0.805
GPT 2-medium-345M	**RF**	0.712	0.740	0.726	0.751	0.723	0.737
	LR	0.751	0.794	0.772	0.799	0.756	0.777
	SVM	0.738	0.799	0.767	0.799	0.738	0.767
GPT 2-large-762M	**RF**	0.706	0.752	0.728	0.731	0.682	0.706
	LR	0.757	0.803	0.779	0.787	0.740	0.763
	SVM	0.760	0.804	0.781	0.789	0.742	0.765
GPT 2-xl-1542M	**RF**	0.747	0.740	0.744	0.745	0.752	0.748
	LR	0.783	0.794	0.789	0.793	0.782	0.787
	SVM	0.780	0.805	0.792	0.800	0.774	0.787
GPT 2-xl-1542M Generated	**RF**	0.724	0.731	0.727	0.726	0.720	0.723
	LR	0.769	0.773	0.771	0.782	0.764	0.773
	SVM	0.762	0.781	0.772	0.770	0.781	0.761
GPT 3.5	**RF**	0.857	0.868	0.863	0.871	0.860	0.865
	LR	0.873	0.920	0.896	0.918	0.870	0.894
	SVM	0.882	0.909	0.895	0.909	0.882	0.896

these values among different models. Notice that among the original GPT2 data sets, GPT2-small has the highest F-measure, meaning it is the easiest to be detected. This result makes sense considering it has the least parameters. However, no regular drops are perceived when the parameters of the model increase, which is different from Solaiman et al. (2019)'s results. This indicates that our system perceives the text generated by GPT2 models of different parameters as very similar. Overall, we achieve performance similar to Solaiman et al. (2019). This is outstanding as Solaiman et al. (2019) use 250,000 data entries for training while our system only uses 5000 entries, 1/50 of their size.

For the text generated by GPT2 pipelined from New York Times, our system still has fairly high performance in differentiating them. However, the overall performance has a slight drop from the Solaiman et al. (2019) dataset. This is probably because to some extent, the filling of our New York Times content makes the text generated by GPT2 more similar to human-written.

Surprisingly, when comparing the results of detecting GPT2-NYT-generated text and GPT3.5-generated data, the performance of the latter is much better (more than 10%). We speculate that one or more of the features are actually differentiating GPT3.5 text efficiently. We perform a further analysis to investigate which features contribute to such results.

We study the statistical feature of the GPT3.5 and observe that feature 2.1.5 Digit Numbers for generated data and written data has a significant difference, as demonstrated in the box plot. Therefore, our hypothesis is that the difference in this feature makes it actually easier to classify the data.

Another notable difference in the result is that Random Forests (RF) in general have worse performance compared to two other models. We speculate that this is because RF is more useful for POS tagging, parsing, and named entity recognition as suggested in Xu and Jelinek (2004). Additionally, RF's ensemble nature, which combines multiple decision trees, can increase complexity and the risk of overfitting, particularly when the available data (10,000 samples in our case) is insufficient for effective training.

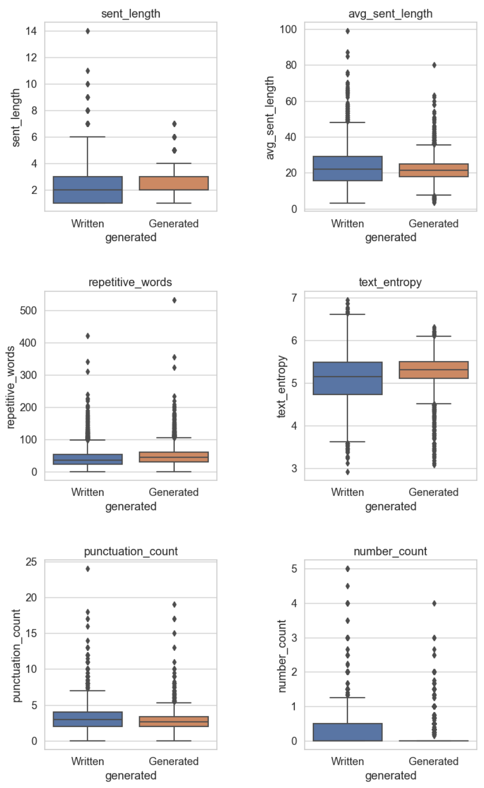
Figure 1: Feature Comparison

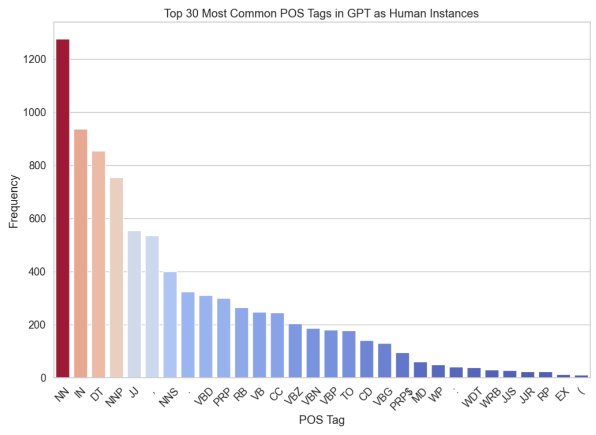
Figure 2: Frequency Dist. of POS Tag: GPT as Human

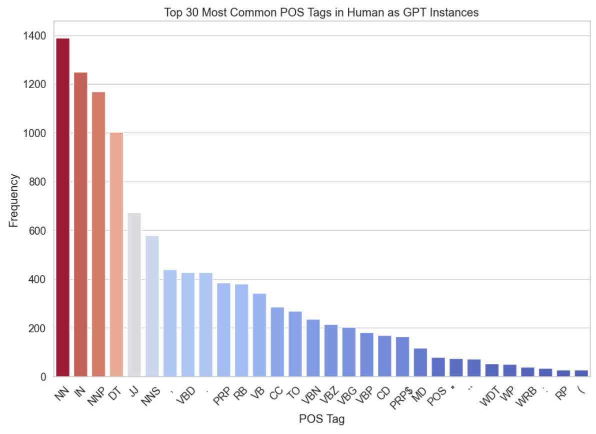
Figure 3: Frequency Dist. of POS Tag: Human as GPT

9 Error Analysis

For our error analysis, we select the dataset from GPT-2-xl-1542m (section 7.2.4) as it has the largest number of parameters. We focus on the results of the SVM model, as it achieves the highest F-measure among the three models, and examine its misclassified instances.

First, we identify instances of GPT 2-xl-1542m data misclassified as human and human data misclassified as GPT 2-xl-1542m data. This is done by comparing the true labels with the predicted labels.

Then, we create two separate datasets for the misclassified instances: gpt_as_human_data for GPT data misclassified as human, and human_as_gpt_data for human data misclassified as GPT.

After that, we analyze the misclassified instances by visualizing the frequency distribution of POS tags.

From the most 30 misclassified POS Tags (Figure 2 and Figure 3), we see that Nouns, Prepositions, determiners, and proper nouns are the four most common Parts of Speech that occur when the model classified GPT data as human text.

10 Conclusion

The main objective of this paper was to investigate the performance of detecting machine-generated text through statistical features. We collected text data from New York Times articles, utilized existing GPT2-text datasets, and generated custom text through GPT models. We extracted the average sentence length, the ratio of repetitive words entropy, punctuation, and digit numbers of excerpts to perform training and classification on logistic regression, random forest, and support vector machine. Trained from 5000 entries, our models achieve similar performances (75%-78%) as Solaiman et al. (2019)'s work trained on 250,000 entries (74%-88%). Additionally, the classification between New York Times original text and GPT2-pipelining text has a small performance drop(2.2%). The classification between New York Times original text and GPT-3.5-generated text has much better performance (88%). This is probably because the text generated by GPT 3.5 from prompts with titles has much less number of words per sentence due to not having enough details to fill the text.

11 Future Work

11.1 Predication without knowing the LLM used

It would be interesting to explore the possibility of training a single model that can detect and classify AI-generated text and identify the specific model used to generate it. This would eliminate the need for multiple models and simplify the classification process. One approach could be to leverage transfer learning techniques and fine-tune a pre-trained language model on a dataset of AI-generated text from multiple models. Another approach could be to use ensemble methods that combine the strengths of different models for better classification performance. Moreover, the exploration of novel features that could distinguish between AI-generated text and human-written text could improve the accuracy and robustness of the classification models. Overall, the potential for a single, highly accurate model for detecting and classifying AI-generated text is a promising direction for future research.

11.2 Other features

Aside from statistical features, there are several other features that could be used to detect AI-generated text, and integrating these features in the model is also another potential.

1. Language models: Language models like GPT-3 are trained on a vast amount of text data and can generate text that resembles human writing. However, they often lack coherence and may generate text that is off-topic or illogical.

2. Tone and style: AI-generated text may lack the nuance and subtlety of human writing. By analyzing the tone and style of a piece of text, it may be possible to determine whether it was generated by an AI or written by a human.

3. Domain-specific knowledge: AI-generated text may lack domain-specific knowledge that is present in the human-written text. By analyzing the content of a piece of text, it may be possible to determine whether it was generated by an AI or written by a human with domain-specific expertise.

4. Common errors: AI-generated text may contain common errors or patterns that are not present in the human-written text. By analyzing the errors and patterns present in a piece of text, it may be possible to determine whether it was generated by an AI or written by a human.

5. Sentiment analysis: AI-generated text may lack the emotional nuances present in the human-written text. By analyzing the sentiment of a piece of text, it may be possible to determine whether it was generated by an AI or written by a human.

Ethics Statement

It can be challenging to identify the definition of machine-generated text and human-generated text when humans and machines collaborate to create writings. There are three scenarios to consider. Firstly, humans compose the text and the AI rewrite it, which makes it difficult to discern whether it is machine-generated or human-written. Secondly, the text is created by AI, and then it is revised by humans, adding another level of intricacy to assign whether humans or machine generate the text. Finally, there are situations where humans type their requirements to the machine continuously and machine generates the text following humans' suggestions.

Acknowledgements

We want to express our sincere gratitude to Professor Adam Meyers. His significant suggestions and feedback played a crucial role in shaping the quality of our research. Additionally, we would like to special thanks to our TA Norihito Naka for helping with model selection and data interpretation. Their insightful advice and counsel have greatly aided us to overcome the difficulties. This research would not have been possible without their guidance.

References

April Dae Bation, Aileen Joan Vicente, and Erlyn Manguilimotan. 2017. Automatic categorization of Tagalog documents using support vector machines. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 346–353. The National University (Phillippines).

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for*

Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 7282–7296, Online. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Jinfen Li, Zhihao Ye, and Lu Xiao. 2019. Detection of propaganda using logistic regression. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Allen Schmaltz. 2018. On the utility of lay summaries and AI safety disclosures: Toward robust, open research oversight. In *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*, pages 1–6, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models.

Peng Xu and Frederick Jelinek. 2004. Random forests in language modelin. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 325–332, Barcelona, Spain. Association for Computational Linguistics.

A Appendix

We build a server in our code with Python Flask that can be deployed easily. Users can put custom data under the data folder and perform training. The machine learning models trained from the training script will be saved into files. Then, the deployed server can take in text requests, fetch the models from storage, perform classification, and return the results to the user. Source code:
`https://github.com/majunze2001/NLP-final-project`