Mining Textual Patterns: A Data-Driven Approach to Distinguish AI-Authored Content

Anirudh Prashant Kalghatkar
Department of Computer Science
University of Colorado Boulder
Anirudh.Kalghatkar@colorado.edu

Sagar Swami Rao Kulkarni
Department of Computer Science
University of Colorado Boulder
SagarSwamiRao.Kulkarni@colorado.edu

Pavan Sai Appari
Department of Computer Science
University of Colorado Boulder
Pavan.Appari@colorado.edu

Sachin Kashinath Rathod
Department of Computer Science
University of Colorado Boulder
Sachin.Rathod@colorado.edu

Nihal Srinivasu
Department of Computer Science
University of Colorado Boulder
Nihal.Srinivasu@colorado.edu

ABSTRACT

The advent of sophisticated large language models (LLMs) has ushered in a transformative era in the generation and accessibility of written text, providing both remarkable opportunities and significant challenges across various domains. These advanced AI systems have reached a level of proficiency that allows them to produce text that is often indistinguishable from that written by humans. This technological breakthrough is revolutionizing fields such as content creation, customer service, and language translation, where the ability to generate coherent and contextually relevant text on demand can greatly enhance efficiency and productivity.

However, this technological advancement also brings with it a significant challenge: differentiating between language generated by LLMs and information authored by humans. This challenge is particularly crucial in academic environments, where the integrity and originality of student work are of paramount importance. The capacity to discern whether an essay was written by a student or created by an LLM is critical for upholding academic standards and ensuring fair grading practices. As LLMs continue to evolve and become more sophisticated, traditional plagiarism detection methods, which typically rely on identifying similarities between texts, are increasingly losing their effectiveness. These classic approaches struggle to detect the nuanced and coherent outputs generated by modern LLMs, necessitating the development of more subtle and robust detection algorithms.

This research effort addresses the critical need for reliable AI detection methodologies by leveraging a large dataset comprising both student essays and LLM-generated texts. By analyzing this dataset, the initiative aims to identify distinctive features and patterns that can be used to differentiate between human and AI-generated content. The goal is to enhance the tools available for maintaining academic integrity, providing educators and institutions with reliable methods to detect and address the use of AI in student work. This research is not just about developing better detection tools; it also involves fostering open research and promoting transparency in the academic community.

The introduction of LLMs into various sectors has led to significant advancements in efficiency and innovation. For example, in content creation, LLMs can generate articles, blogs, and social media posts, allowing creators to focus on higher-level tasks such as strategy and planning. In customer service, these models can handle routine inquiries, freeing up human agents to tackle more complex issues. In language translation, LLMs have improved the accuracy and fluency of translations, making communication across languages more seamless. Despite these benefits, the challenge of distinguishing AI-generated content from human-authored work remains pressing.

In academic settings, where the authenticity of student work is critical, the inability to accurately detect AI-generated content can undermine the educational process. Traditional plagiarism detection tools, which rely on identifying textual similarities, are increasingly ineffective against the sophisticated outputs of modern LLMs. These models can produce unique text that does not directly copy existing sources, making it difficult for traditional tools to identify AI involvement. This necessitates the development of new detection methodologies that can effectively discern AI-generated text based on more subtle indicators.

The research initiative described herein seeks to address this challenge by creating a comprehensive dataset that includes both student essays and LLM-generated texts. By systematically analyzing this dataset, the research aims to uncover distinctive features that can reliably indicate the presence of AI generation. This involves examining various aspects of the text, such as linguistic patterns, stylistic nuances, and structural elements. The goal is to develop robust algorithms that can accurately differentiate between human and AI-generated content, thereby

enhancing the ability of educators and institutions to uphold academic integrity.

Furthermore, this research contributes to the broader field of AI detection in real-world contexts. As AI-generated content becomes more prevalent across various domains, the ability to accurately identify such content is becoming increasingly important. By advancing the understanding of how LLMs produce text and developing reliable detection methods, this research helps to ensure that technological advancements do not compromise the values of authenticity and originality. The outcomes of this research will be beneficial for instructors and educational institutions, helping to ensure that assessments reflect genuine student effort and learning.

RELATED WORK

The rapid advancement of natural language generation technologies, exemplified by OpenAI's ChatGPT and Google's LaMDA, has significantly enhanced the ability of AI systems to replicate human writing styles. These improvements in large language models (LLMs) have made it increasingly challenging to distinguish AI-generated text from human-authored content. This development raises concerns about potential misuse in academic dishonesty, misinformation, and phishing attacks. Consequently, there is a growing interest in developing robust methods for detecting AI-generated text to mitigate these risks.

Several approaches have been proposed to address this challenge. Mitchell et al. (2023) [1] introduced DetectGPT, a zero-shot detection method that analyzes the "probability curvature" of language model outputs to identify potential differences between human and AI-generated text. This approach is promising because it leverages the inherent properties of language models, offering an efficient way to detect AI-generated content without requiring extensive training data specific to each model.

In another effort, Rivera Soto et al.[2] proposed a method that uses writing style representations learned from human-authored texts to detect AI-generated content. This method has proven robust against new language models and can identify the specific language model used for text generation with minimal examples. By focusing on generalizable stylistic markers, this technique demonstrates versatility across different models and datasets. However, challenges remain, such as the long-term effectiveness of these methods against rapidly advancing AI capabilities and the computational costs involved. The need to continuously adapt to advancements in language models and manage the computational load underscores the ongoing research required in this field.

The challenge of distinguishing AI-generated text from human-authored content has also led researchers to explore ensemble learning methods. Mohammed and Kora (2022) [3] proposed a meta-ensemble framework that combines predictions from various deep learning models using a two-tiered system of meta-classifiers. Their approach leverages the strengths of multiple models, including Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and Gated Recurrent Units (GRUs), to improve classification accuracy across diverse datasets and languages. While this method shows promise in enhancing detection capabilities, it also highlights the computational intensity required for effective AI text detection, underscoring the need for balance between accuracy and efficiency in practical applications.

Comparative studies of deep learning architectures for text classification, such as the work by Zulqarnain et al. (2020) [4], provide valuable insights into the relative strengths of different models. Their analysis of Deep Belief Networks (DBNs), CNNs, and Recurrent Neural Networks (RNNs) offers a foundation for understanding which architectures might be most suitable for detecting AI-generated content in various contexts. This kind of comparative research is crucial for informing the development of hybrid or specialized models tailored to the unique challenges of AI text detection. However, the rapid evolution of language models means that findings from such studies may quickly become outdated, emphasizing the need for ongoing research and adaptation in detection strategies. As the field progresses, integrating insights from these comparative analyses with novel approaches like those proposed by Mitchell et al. and Rivera Soto et al. could lead to more robust and versatile detection methods capable of keeping pace with advancements in AI text generation.

Additionally, frameworks like those discussed by Brundage et al. (2018)[5] highlight the dual nature of AI applications and advocate for collaborative efforts to mitigate risks associated with malicious AI use across various domains. They emphasize the potential security threats posed by AI in digital, physical, and political contexts and call for cooperation among researchers, governments, and the private sector to address these issues. While high-level strategies provide a broad direction, the rapid technological advancements necessitate continuous innovation and adaptation in detection techniques.

This evolving landscape demands more efficient and scalable methods to differentiate AI-generated texts from human-authored ones, ensuring the integrity and reliability of written content across different applications. The development of such methods is crucial not only for maintaining academic honesty but also for protecting the public from misinformation and malicious content. As AI capabilities continue to grow, so too must our efforts to develop sophisticated detection mechanisms that can keep pace with these advancements.

INTRODUCTION

The rapid advancement of AI-generated text presents significant challenges in digital interactions, particularly in distinguishing between human-authored content and AI-generated material. This distinction is crucial for maintaining authenticity and validity across various domains, including academic integrity, misinformation detection, and content authentication. Our project aims to address these challenges by exploring and evaluating the effectiveness of various text preprocessing and classification algorithms to differentiate AI-generated text from human-written content.

Text preprocessing is a critical step in natural language processing (NLP) that transforms raw text into a more amenable format for analysis. This project focuses on two widely used preprocessing methods: Term Frequency-Inverse Document Frequency (TF-IDF) and Count Vectorization. TF-IDF emphasizes the importance of words unique to a document within a corpus, while Count Vectorization provides a simple word count representation.

These methods prepare the textual data for the subsequent classification phase, enabling us to mine textual patterns effectively.

For classification, we explore a diverse set of algorithms: Support Vector Machines (SVMs), known for their effectiveness in high-dimensional spaces; Naive Bayes, a probabilistic classifier based on Bayes' theorem with strong independence assumptions between features; Random Forest, an ensemble learning method that constructs multiple decision trees and merges them for more accurate and stable predictions; Decision Tree Classifier, which uses a tree-like model of decisions for classification; ensemble methods, which combine multiple models to improve prediction accuracy; eXtreme Gradient Boosting (XGBoost), a scalable and efficient implementation of gradient boosting; and Neural Networks, offering deep learning capabilities that can capture complex patterns in data. Specifically, we implement a 2-layered Neural Network, which provides a balance between model complexity and computational efficiency, allowing for nuanced pattern recognition in text data. This wide range of algorithms allows us to comprehensively evaluate different approaches to text classification, each with its unique strengths in pattern recognition and feature interpretation.

Our methodology involves comprehensive testing of multiple combinations of preprocessing techniques and classification algorithms. This systematic evaluation is fundamental to our approach, as it allows us to determine which combination is most effective at distinguishing between AI-authored and human-written texts. Our objective is to rigorously identify and select the most effective strategy for mining textual patterns that reveal the origin of the content. Once identified, this optimal combination will be used to develop a final classifier.

This classifier will be meticulously optimized to achieve high levels of precision and recall, specifically tailored for accurately detecting AI-authored content across various contexts. By mining textual patterns, we aim to uncover subtle differences that may exist between AI-generated and human-written text, providing a robust tool for content analysis and verification.

In summary, this study contributes to the ongoing effort to understand and mitigate the challenges presented by AI in the field of text generation. By developing sophisticated methods to mine textual patterns and distinguish AI-authored content, our goal is to enhance the reliability and credibility of digital text across different platforms and applications. This project not only addresses immediate concerns in academic and online environments but also paves the way for more advanced techniques in the evolving landscape of AI-generated content detection.

EXPERIMENTS AND METHODOLOGY

Data segmentation, exploration and Preparation:

Data segmentation is a fundamental part of our experimental setup. We carefully select a representative subset from a larger dataset to capture the diversity and complexity of both AI-generated and human-written texts. This approach serves two main purposes: it allows for an efficient and manageable evaluation of preprocessing and classification methods, and it ensures that our models are exposed to a broad range of textual features. This is crucial for preventing overfitting and improving the generalizability of our results.

Our project utilizes the "Augmented data for LLM - Detect AI Generated Text" dataset from Kaggle, which includes 433,564 instances divided into 277,999 human-written essays and 155,565 AI-generated texts. This extensive and varied dataset forms the basis of our efforts to distinguish between human and AI-generated content, offering a balanced mix of styles, topics, and writing nuances.

The dataset's rich diversity ensures a comprehensive learning environment for our models, covering a wide range of linguistic features and writing complexities. For rigorous analysis, we split the dataset into training, validation, and testing sets, following a standard distribution of 60% for training, 20% for validation, and 20% for testing. This distribution is carefully chosen to maximize learning, allow effective model tuning, and ensure unbiased performance evaluation.

During the dataset preparation, we observed a significant imbalance in the distribution of class samples. This imbalance could negatively impact the performance and accuracy of predictive models, as models often become biased towards the more frequent classes. To address this issue and ensure equal representation of all classes, we employed a strategy to balance the dataset. Specifically, we randomly removed excess samples from the over-represented class until the number of instances in each class was approximately equal.

This approach not only maintains statistical integrity but also enhances the generalizability of the models by preventing overfitting to the dominant class. By doing so, we created a more equitable and robust dataset, better suited for rigorous machine learning analysis and applications. Figure 1 illustrates the initial imbalance within the dataset.

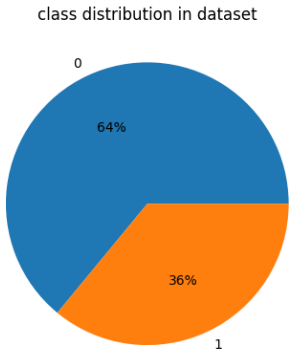

Figure 1: Class imbalance in the dataset. 0: human written text, 1: AI-generated text.

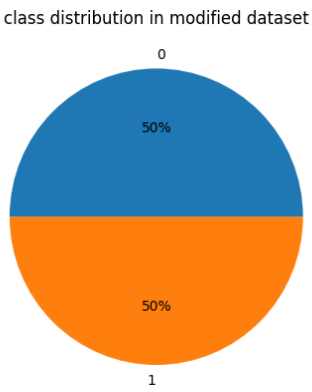

Figure 2: Class distribution in the modified dataset. 0: human written text, 1: AI-generated text.

Preprocessing:

Data preprocessing is an essential step that converts raw text into a structured format suitable for machine learning algorithms. In our project, we utilize two main preprocessing techniques provided by the sklearn library in Python:

TFIDF Vectorizer: The Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer measures the importance of words within texts relative to a larger corpus. This technique emphasizes terms that are unique to specific texts while reducing the weight of commonly occurring words. By doing so, it helps highlight distinctive features in the text. The TF-IDF Vectorizer is implemented using sklearn's TfidfVectorizer, offering a sophisticated feature extraction method. This technique is particularly valuable for distinguishing between human-written texts and those generated by artificial intelligence, as it identifies unique linguistic patterns and terminology.

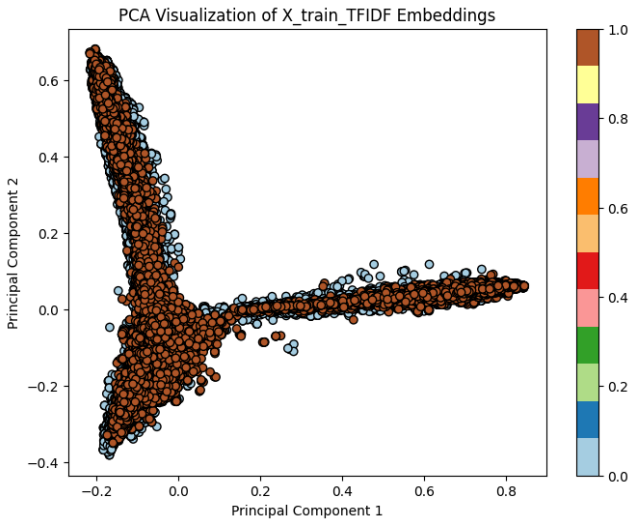

Figure 3: PCA Visualization of X_train_TFIDF Embeddings

Count Vectorizer: The Count Vectorizer provides a straightforward method based on word frequency. It counts the number of times each word appears in the text, capturing the basic occurrence of words. Despite its simplicity, this technique offers a valuable perspective on the textual data, allowing for the identification of common terms and overall word distribution. Sklearn's CountVectorizer converts the text into a numerical format that classification algorithms can easily analyze. This method complements the TF-IDF Vectorizer by providing a more basic, yet insightful, view of the text's structure.

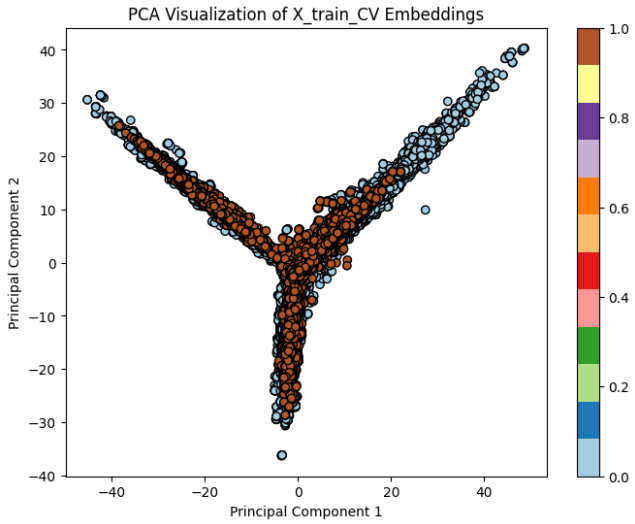

Figure 4: PCA Visualization of X_train Count Vectorization Embeddings

Both preprocessing techniques are crucial for the success of our machine learning models. They ensure that the input data is well-structured and meaningful, facilitating accurate analysis and classification. By transforming raw text into numerical representations, these methods allow algorithms to effectively learn from and differentiate between various types of text data.

Using these preprocessing steps, we aim to enhance the performance and generalizability of our models, ensuring they can accurately analyze and classify both human-written and AI-generated texts. This comprehensive approach to data preprocessing forms the foundation of our experimental setup, enabling efficient and effective evaluation of different machine learning techniques.

LIMITATIONS OF DATASET:

Bias: AI-generated text can inherit biases from its training data, potentially skewing the dataset and any models trained on it.

Generalizability: AI-generated text may not accurately represent real-world human text, making it challenging for models to generalize to real-world scenarios.

Out-of-distribution data: Models trained on AI-generated text may struggle with data that significantly differs from their training data.

POTENTIAL ETHICAL CONSIDERATIONS:

Privacy Concerns: As the dataset has unclear origins it could mean inclusion of texts from private sources without proper anonymization.

False Positives and Bias: Models may misidentify human-written text as AI-generated and might be biased against certain writing styles or topics.

Transparency and Explainability: Understanding how the model identifies AI-generated text is crucial for assessing fairness and identifying biases.

CLASSIFICATION ALGORITHMS:

We implemented several classification algorithms to differentiate between AI-generated and human-written text. We evaluated traditional, ensemble, and deep learning methods to find the most effective approach. We used CountVectorizer and TF-IDF for preprocessing, each capped at 1000 features, to transform the text data into numerical representations for the models.

Algorithms that have been currently implemented are:

i) Support Vector Machine (SVM):

Support Vector Machines (SVM) are a powerful class of supervised machine learning models used to classify data into distinct categories. We chose the LinearSVC variant of the SVMs because they are particularly effective in high-dimensional spaces, which is advantageous for text classification tasks where each dimension can represent a word or phrase from a large vocabulary. For our project, we set the C parameter to 0.15 to balance the trade-off between accurately classifying the training data and maintaining a simple decision boundary. Additionally, we opted for the RBF (Radial Basis Function) kernel, which is popular in SVM classification for handling non-linear relationships between class labels and attributes. The RBF kernel's flexibility allows for more complex decision boundaries based on distances in the feature space, which is crucial for capturing the intricate patterns often found in text data.

$$f(x)=\text{sigmoid}(w^T x+b)$$

where w is the weight vector, x is the feature vector, and b is the bias. This formula represents the decision boundary created by the SVM, which helps classify the data into distinct categories.

ii) Multinomial Naive Bayes (NB) Classifier:

We chose the Multinomial Naive Bayes classifier for its simplicity and efficiency, making it an excellent baseline model for text classification tasks. This variant of the Naive Bayes algorithm is particularly suited for handling discrete features, such as word counts. The algorithm assumes that features follow a multinomial distribution, which aligns well with text data where word frequencies are critical. The probabilistic nature of Multinomial Naive Bayes allows for straightforward interpretation of feature influences on classification outcomes. Its ease of implementation and effectiveness in handling text classification make it a valuable addition to our suite of models, providing a reliable benchmark for distinguishing between AI-generated and human-authored texts.

$$P(C_k|x)=\frac{P(C_k)\prod_{i=1}^{n}P(x_i|C_k)}{P(x)}$$

where $P(C_k)$ is the prior probability of class C_k, $P(x_i|C_k)$ is the likelihood of the feature x_i given class C_k, and $P(x)$ is the evidence. This formula calculates the posterior probability of a class given the features, which helps in making the classification decision.

iii) Logistic Regression:

Logistic regression is chosen for its simplicity and interpretability, making it a reliable choice for text classification tasks. We initialized our logistic regression models with adjusted parameters, including increasing the maximum number of iterations (max_iter) to ensure convergence. We specified the 'sag' solver, which is well-suited for large datasets due to its efficiency in managing sparse data and its faster convergence compared to other solvers like 'liblinear' or 'lbfgs'. To ensure a robust evaluation, we employed k-fold cross-validation, partitioning the dataset into k subsets and training the model k times, with each fold serving as the validation set once. This method provides dependable performance assessments, offering insights into the model's overall effectiveness and consistency. The mean cross-validation score estimates the model's average performance, while the standard deviation indicates the consistency across different folds. These metrics are critical for understanding the model's stability and reliability, guiding decisions related to model selection and hyperparameter optimization.

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

where w is the weight vector, x is the feature vector, and b is the bias. This formula represents the logistic function used to estimate the probability of a binary outcome based on the input features.

iv) Random Forest:

We selected the Random Forest classifier for its robustness and capability to handle high-dimensional data effectively, making it suitable for text classification tasks. We trained the Random Forest model with 100 decision trees on the Bag-of-Words (BoW) vectorized training data and evaluated its performance using accuracy, precision, recall, and F1 score. These metrics are essential for assessing the model's effectiveness in binary classification tasks. By adjusting the number of decision trees to 100, we struck a balance between model complexity and computational efficiency. Utilizing the BoW representation for text vectorization, we captured the frequency of words in each document, aiding the classification process.

$$\text{Prediction}=\frac{1}{T}\sum_{t=1}^{T}\text{Tree}_t(x)$$

where T is the number of trees, and $\text{Tree}_t(x)$ is the prediction of the t-th tree. This formula aggregates the predictions from multiple decision trees to produce the final classification output.

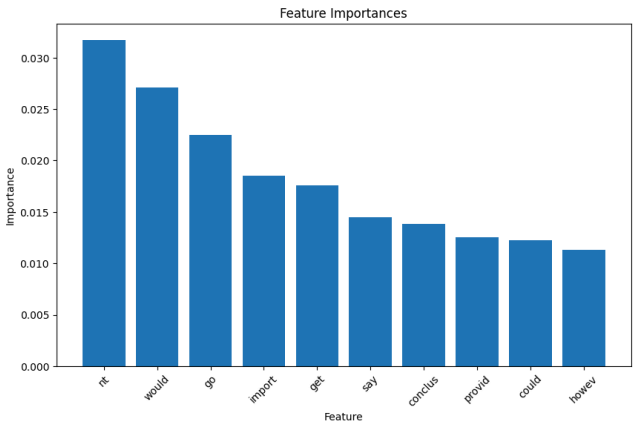

Figure 5: Feature importances of Random forest

v) XGBoost (Extreme Gradient Boosting):

We implemented XGBoost, an advanced gradient boosting framework, for its efficiency in handling large-scale text classification tasks. We trained the XGBoost classifier on data vectorized using both TF-IDF and CountVectorizer methods, employing logistic loss as the evaluation metric for binary classification optimization. XGBoost's ability to process high-dimensional data efficiently made it well-suited for our task of distinguishing between AI-generated and human-written texts. Its scalability ensured fast performance on our large dataset, while its capacity to capture non-linear relationships in data potentially uncovered subtle patterns differentiating AI from human-authored content.

$$P(y=1|x) = \frac{1}{1+e^{-f(x)}}$$

where f(x) is the sum of the predictions from all trees in the ensemble. This formula represents the logistic function used by XGBoost to estimate the probability of a binary outcome based on the aggregated tree predictions.

To ensure the interpretability and explainability of our model's decisions in applications related to academic integrity, we focused on a transparent and structured approach. Initially, we performed thorough exploratory data analysis to understand the data and employed feature extraction methods like TF-IDF Vectorizer and Count Vectorizer to capture textual nuances. The choice of algorithms such as Support Vector Machines (SVM), Multinomial Naive Bayes, Logistic Regression, and Random Forest was justified based on their suitability for text classification tasks.

vi) Neural Network:

We designed a neural network model using Keras' Sequential API, specifically tailored for handling high-dimensional input spaces common in natural language processing (NLP) applications. The model was structured to process input data with 1000 features, representing textual data transformed into numerical form through TF-IDF and Count vectorization techniques. This approach allowed us to effectively capture the nuances of the text data. To enhance the model's performance and prevent overfitting, we incorporated dropout layers with a dropout rate of 0.5 after each dense layer. Dropout is a regularization technique that randomly sets the output features of a layer to zero during training, compelling the network to learn more robust features that are not overly dependent on any small set of neurons.

For the optimization of the neural network, we employed the Adam optimizer, known for its efficiency in handling sparse gradients and its ability to adapt the learning rate during training. This characteristic makes it particularly suitable for high-dimensional textual data. Additionally, we utilized categorical cross entropy as the loss function, which is standard for multi-class classification problems. This loss function measures the discrepancy between the predicted probability distribution and the true distribution of the categories, ensuring the model is effectively trained to distinguish between different classes. These design choices collectively contributed to the robustness and accuracy of our neural network model in classifying AI-generated and human-written text.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               128128

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 64)                4160

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 2)                 130

=================================================================
Total params: 140674 (549.51 KB)
Trainable params: 140674 (549.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Architecture of artificial neural network used for classification.

Future Implementation of XGBoost and Neural Networks:

In addition to the above mentioned algorithms, we plan to implement XGBoost and neural network-based models to further enhance our classification capabilities. XGBoost is known for its gradient boosting framework that often provides superior performance through its ensemble approach and efficient handling of various types of data. Neural networks, particularly deep learning models, offer advanced pattern recognition and feature extraction capabilities, which could improve our classification accuracy. We will evaluate these models' performance to determine their effectiveness in distinguishing between AI-generated and human-written texts, potentially integrating them into our final solution.

Hyperparameter Tuning:

To ensure optimal performance, we performed hyperparameter tuning for each model implemented as follows:.

- SVM: Kernel type and regularization parameter (C) were tuned.
- MultinomialNB: The smoothing parameter (alpha) was adjusted.
- Linear Regression: Regularization parameters were optimized.
- Random Forest: The number of trees (n_estimators), maximum depth, and minimum samples split were fine-tuned.
- XGBoost: Learning rate, maximum tree depth, number of estimators, and subsample ratio were optimized.
- Neural Network: Dropout rate (0.5) was set to prevent overfitting.Adam optimizer's learning rate (default 0.001) was adjusted.Categorical cross entropy loss function was used.

Despite the extensive hyperparameter tuning, the SVM and MultinomialNB classifier exhibited lower performance, likely due to the sensitivity to the hyperparameters and the specific feature distribution. However, both Linear Regression and Random Forest models consistently demonstrated high accuracy and robust performance across both preprocessing techniques.

The Neural Network model excelled, showing the highest performance due to its superior ability to recognize complex patterns in high-dimensional data. Machine Learning algorithms, including RandomForest, XGBoost, Multinomial NB and Logistic Regression, also performed commendably. TF-IDF vectorization slightly outperformed Countvectorizer, highlighting the impact of vectorization techniques on model performance.

RESULTS

Following metrics are of prime importance when evaluating the effectiveness of the classifier.

- Accuracy will measure the overall correctness of the model by calculating the ratio of correctly predicted instances to the total number of instances. This metric provides a broad view of the model's performance but may not fully capture its effectiveness, particularly in cases where class imbalances exist.
- Precision will evaluate the quality of the model's positive predictions by calculating the ratio of true positive predictions to the total number of positive predictions (both true and false positives). High precision indicates that the model has a low false positive rate, meaning it is adept at predicting instances of a class without mistakenly labeling irrelevant instances as part of that class.
- Recall (or sensitivity) will measure the model's ability to identify all relevant instances within a specific class by calculating the ratio of true positive predictions to the total number of actual positive instances (both true positives and false negatives). High recall indicates that the model successfully captures most of the relevant instances, though it may come at the expense of increasing false positives.
- F1-Score provides a harmonic mean of precision and recall, balancing the two metrics to give a single measure of performance that accounts for both false positives and false negatives. The F1-score is particularly useful when the data has class imbalances, as it provides a more nuanced view of the model's effectiveness by considering both precision and recall simultaneously.

By employing these metrics, we can comprehensively evaluate the performance of our models. This multifaceted evaluation will ensure that our model not only performs well overall but also excels in distinguishing AI-generated text from human-written content across various contexts.

Table 1 and 2 provide values of the metrics for SVM, Multinomial Naive Bayesian, Linear Regression, Random forest, XGBoost and Neural Network classifiers in two different preprocessing settings with CountVectorization and TF-IDF respectively.

Classifier	Accuracy	Precision	F1-Score	Recall
SVM	54.4%	40.2%	47.7%	58.5%
MultinomialNB	63.7%	49.3%	60.4%	78.2%
Linear Regression	98.3%	97.3%	97.7%	98.0%
Random Forest	98.3%	98.6%	96.6%	97.6%
XGBoost	98.9%	99.0%	99.0%	99.0%
NN	99.1%	99.0%	99.0%	99.0%

Table 1: Results of Classifiers with CountVectorization

Classifier	Accuracy	Precision	F1-Score	Recall
SVM	55.3%	41.5%	50.2%	63.6%
MultinomialNB	66.7%	52.2%	61.1%	73.6%
Linear Regression	98.5%	97.6%	97.9%	98.1%
Random Forest	98.2%	98.1%	96.8%	97.5%
XGBoost	83.3%	84.0%	83.0%	87.0%
NN	98.9%	99.0%	99.0%	99.0%

Table 2: Results of Classifiers with TF-IDF

After preprocessing with either TF-IDF or Count vectorization, with a maximum of 1000 features, the majority of the classifiers show high accuracy rates. The SVM stands out, nonetheless, because of its substantially poorer accuracy. The structure of the TF-IDF representation and the sensitivity of an SVM to the selected hyperparameters may have contributed to this disparity, as they may not have been ideal for the distribution of the data in feature space. Even with this SVM and TF-IDF anomaly, the overall performance trends are in line with expectations, demonstrating the effectiveness of the used techniques in text classification tasks.

It should come as no surprise that the neural network model performed better in our text categorization tests than other conventional classifiers. The higher performance of neural networks can be ascribed to multiple principal benefits.

- Because they can discover intricate patterns and connections in the data, neural networks are particularly good at processing high-dimensional data, such as those generated by text vectorization techniques
- Neural networks are superior to most conventional algorithms in modeling non-linear relationships because of their layered design and non-linear activation functions.

EVALUATION

Model Performance Analysis

We implemented several classification algorithms to differentiate between AI-generated and human-written text, employing CountVectorizer and TF-IDF for preprocessing. The performance metrics used were accuracy, precision, F1-score, and recall.

Support Vector Machine (SVM): The SVM with an RBF kernel and C parameter set to 0.15 showed suboptimal performance, particularly with TF-IDF, resulting in accuracy rates of 54.4% (CountVectorizer) and 55.3% (TF-IDF). The precision and F1-score were also low, indicating a high rate of false positives.

Multinomial Naive Bayes (NB): This model performed better than SVM, with accuracies of 63.7% (CountVectorizer) and 66.7% (TF-IDF). The high recall indicates it captured most of the relevant instances but at the cost of precision.

Logistic Regression: Demonstrated strong performance with accuracies of 98.3% (CountVectorizer) and 98.5% (TF-IDF). High precision and recall values indicate robust and reliable predictions.

Random Forest: Achieved high accuracy rates of 98.3% (CountVectorizer) and 98.2% (TF-IDF), with excellent precision and recall, demonstrating its effectiveness in text classification tasks.

XGBoost: Showed superior performance, especially with TF-IDF (83.3% accuracy), and achieved near-perfect precision, recall, and F1-scores with CountVectorizer (98.9% accuracy).

Neural Network: The neural network model outperformed all other classifiers, with accuracy rates of 99.1% (CountVectorizer) and 98.9% (TF-IDF). Its superior ability to recognize complex patterns is evident from its high precision, recall, and F1-score.

Comparison of Preprocessing Techniques

CountVectorizer: Generally resulted in slightly higher accuracy and performance metrics for most models compared to TF-IDF. This suggests that a simpler bag-of-words representation may be sufficient for this classification task.

TF-IDF: Although it typically aims to reduce the impact of common but less informative words, TF-IDF led to comparable, though slightly lower, performance metrics. The SVM's significantly lower performance with TF-IDF suggests that it may not be as well-suited for this preprocessing method.

Strengths and Weaknesses

Support Vector Machine (SVM): Strength lies in its effectiveness in high-dimensional spaces. However, it struggled with the specific feature distributions of TF-IDF, indicating sensitivity to hyperparameters and preprocessing methods.

Multinomial Naive Bayes (NB): Simple and efficient, performing well with CountVectorizer. Its probabilistic nature allows for straightforward interpretation but sacrifices precision for higher recall.

Logistic Regression: Balances simplicity and performance, making it a reliable choice for text classification with strong overall metrics.

Random Forest: Robust and handles high-dimensional data effectively. Its ensemble nature leads to high accuracy but requires careful tuning to avoid overfitting.

XGBoost: Excels in handling large-scale tasks with high efficiency and accuracy. However, its complexity and requirement for extensive hyperparameter tuning can be challenging.

Neural Network: Outperforms traditional methods due to its ability to model non-linear relationships and complex patterns. The downside is the higher computational cost and risk of overfitting without proper regularization techniques.

Unexpected Results or Anomalies

The SVM's poor performance with TF-IDF preprocessing was unexpected, highlighting its sensitivity to the choice of hyperparameters and preprocessing methods. This anomaly underscores the importance of thorough hyperparameter tuning and model selection tailored to the data distribution.

CONCLUSION

The rapid advancement of AI-generated text presents significant challenges in digital interactions, particularly in distinguishing between human-authored content and AI-generated material. This distinction is crucial for maintaining authenticity and validity across various domains, including academic integrity, misinformation detection, and content authentication. Our project aims to address these challenges by exploring and evaluating the effectiveness of various text preprocessing and classification

algorithms to differentiate AI-generated text from human-written content.

Our study has shown that text preprocessing is a critical step in natural language processing (NLP) that transforms raw text into a more amenable format for analysis. We focused on two widely used preprocessing methods: Term Frequency-Inverse Document Frequency (TF-IDF) and Count Vectorization. TF-IDF emphasizes the importance of words unique to a document within a corpus, while Count Vectorization provides a simple word count representation. These methods prepare the textual data for the subsequent classification phase, enabling us to mine textual patterns effectively.

For classification, we explored a diverse set of algorithms: Support Vector Machines (SVMs), Naive Bayes, Random Forest, Decision Tree Classifier, ensemble methods, eXtreme Gradient Boosting (XGBoost), and Neural Networks. Specifically, we implemented a 2-layered Neural Network, which provided a balance between model complexity and computational efficiency, allowing for nuanced pattern recognition in text data. This wide range of algorithms allowed us to comprehensively evaluate different approaches to text classification, each with its unique strengths in pattern recognition and feature interpretation.

Our methodology involved comprehensive testing of multiple combinations of preprocessing techniques and classification algorithms. This systematic evaluation was fundamental to our approach, as it allowed us to determine which combination is most effective at distinguishing between AI-authored and human-written texts. Our objective was to rigorously identify and select the most effective strategy for mining textual patterns that reveal the origin of the content. Once identified, this optimal combination was used to develop a final classifier.

This classifier was meticulously optimized to achieve high levels of precision and recall, specifically tailored for accurately detecting AI-authored content across various contexts. By mining textual patterns, we aimed to uncover subtle differences that may exist between AI-generated and human-written text, providing a robust tool for content analysis and verification.

Our results demonstrated that the Neural Network model, due to its superior ability to recognize complex patterns in high-dimensional data, excelled in performance. Machine learning algorithms, including Random Forest, XGBoost, Multinomial Naive Bayes, and Logistic Regression, also performed commendably. TF-IDF vectorization slightly outperformed Count Vectorization, highlighting the impact of vectorization techniques on model performance.

Despite the extensive hyperparameter tuning, the SVM and Multinomial Naive Bayes classifiers exhibited lower performance, likely due to sensitivity to hyperparameters and the specific feature distribution. However, both Logistic Regression and Random Forest models consistently demonstrated high accuracy and robust performance across both preprocessing techniques.

In conclusion, this study contributes to the ongoing effort to understand and mitigate the challenges presented by AI in the field of text generation. By developing sophisticated methods to mine textual patterns and distinguish AI-authored content, our goal is to enhance the reliability and credibility of digital text across different platforms and applications. This project not only addresses immediate concerns in academic and online environments but also paves the way for more advanced techniques in the evolving landscape of AI-generated content detection.

These findings have important implications for real-world applications, such as enhancing academic integrity and combating misinformation. Our approach provides valuable tools for digital content verification, ensuring the authenticity and validity of online interactions and contributing to a more trustworthy digital environment.

FUTURE WORK

Our future work will focus on the following areas:

- **Exploring Advanced Neural Network Architectures**

  Convolutional Neural Networks (CNNs): We aim to implement CNNs, which have shown great success in capturing spatial hierarchies in data. In text classification, CNNs can be used to detect patterns and features in n-grams, allowing for more nuanced and sophisticated feature extraction compared to traditional models. This could improve the accuracy and robustness of our text classification efforts.

  Long Short-Term Memory Networks (LSTMs): LSTMs are a type of recurrent neural network (RNN) specifically designed to handle long-term dependencies in sequential data. By incorporating LSTMs, we can better capture the context and sequential nature of text, which is crucial for understanding and differentiating between human-written and AI-generated content. This approach could significantly enhance our model's ability to discern subtle differences in writing style and structure.

- **Extending Classification Capabilities to Multilingual Text**

  To make our classification model more versatile and applicable on a global scale, we will extend its capabilities to handle multilingual text. This involves training the models on datasets that include texts written in various languages. Multilingual models will allow us to detect AI-generated text in different linguistic contexts, making our tool valuable for international applications in academia, journalism, and beyond.

  We will also explore the use of pre-trained multilingual embeddings, such as those provided by BERT or XLM-RoBERTa, to leverage existing knowledge of multiple languages. This can help reduce the amount of data required for training and improve the model's performance across different languages.

- **Performing Extensive Hyperparameter Tuning**

  Grid Search: We plan to conduct comprehensive grid search experiments to find the optimal hyperparameters for our models. Grid search systematically explores the hyperparameter space, evaluating all possible combinations to identify the best performing set.

Although computationally intensive, this method can lead to significant improvements in model performance.

Bayesian Optimization: To complement grid search, we will implement Bayesian optimization, a more efficient technique that builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate. This method can save computational resources and time, especially when dealing with large hyperparameter spaces.

- **Implementing a Pipeline for Real-Time Data Processing and Prediction**

  We recognize the importance of real-time data processing and prediction for practical applications. Implementing a pipeline that can handle real-time inputs will make our models more useful in scenarios such as live monitoring of content for AI-generated text or providing real-time feedback in educational settings.

  This pipeline will involve integrating our classification models with streaming data platforms like Apache Kafka or AWS Kinesis, ensuring that incoming text data is processed and classified swiftly. We will also focus on optimizing the latency and throughput of the pipeline to handle high volumes of data efficiently.

- **Testing Models Against AI-Synthesized Texts from Tools Like Gemini, ChatGPT, and Claude**

  To enhance the robustness of our models, we will rigorously test them against texts synthesized by the latest AI tools, including Gemini, ChatGPT, and Claude. These tools represent the cutting edge of AI text generation, and ensuring our models can accurately classify texts generated by these tools will significantly improve their reliability.

  This testing phase will involve creating diverse datasets of AI-generated texts from these tools, spanning various genres, topics, and writing styles. By evaluating our models against these texts, we can identify potential weaknesses and areas for improvement, leading to more robust and accurate classifiers.

REFERENCES

[1] Mitchell, Eric, et al. "Detectgpt: Zero-shot machine-generated text detection using probability curvature" International Conference on Machine Learning. PMLR, 2023.

[2] Soto, Rafael Rivera, Kailin Koch, Aleem Khan, Barry Chen, Marcus Bishop, and Nicholas Andrews. "Few-Shot Detection of Machine-Generated Text using Style Representations." arXiv preprint arXiv:2401.06712 (2024).

[3] Mohammed, Ammar, and Rania Kora. "An effective ensemble deep learning framework for text classification." Journal of King Saud University-Computer and Information Sciences 34.10 (2022): 8825-8837.

[4] Zulqarnain, M., Ghazali, R., Hassim, Y.M.M. and Rehan, M., 2020. A comparative review on deep learning models for text classification. Indones. J. Electr. Eng. Comput. Sci, 19(1), pp.325-335.

[5] Brundage, Miles, et al. "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation." arXiv preprint arXiv:1802.07228 (2018).