# How to Detect AI-Generated Texts?

**Conference Paper** · October 2023

DOI: 10.1109/UEMCON59035.2023.10316132



**3 authors:**

Trung Nguyen
Winona State University
**15** PUBLICATIONS **82** CITATIONS
SEE PROFILE

Amartya Hatua
Fidelity Investments
**17** PUBLICATIONS **87** CITATIONS
SEE PROFILE

Andrew H. Sung
University of Southern Mississippi
**27** PUBLICATIONS **321** CITATIONS
SEE PROFILE

# How to Detect AI-Generated Texts?

Trung T. Nguyen*
*Department of Computer Science*
*Winona State University*
Winona, MN 55987, USA
trung.nguyen@winona.edu

Amartya Hatua*
*AI Center of Excellence*
*Fidelity Investments*
Boston, MA 02210, USA
amartya.hatua@fmr.com

Andrew H. Sung
*School of Computing Sciences and Computer Engineering*
*The University of Southern Mississippi*
Hattiesburg, MS 39401, USA
andrew.sung@usm.edu

***Abstract*—Recent advances in large language models (LLMs) have significantly improved the quality of synthetic text data. LLMs imitate human writing patterns to produce highly natural text, raising serious ethical, moral, legal, social, and economic concerns in various industries. To address these issues, we present a method to distinguish synthetically generated text (SGT) from human-written text (HWT). Our method includes methods for dataset creation, feature engineering, dataset comparison, and result analysis. As part of our research, we created two datasets - a Wikipedia-based dataset and a US Election 2024 related news article-based dataset using ChatGPT. These datasets can be used as open-source datasets in future studies. We also identified a set of handcrafted features that can serve as the baseline feature set for future research.**

***Index Terms*—LLM, ChatGPT, feature engineering, document similarity**

## I. Introduction

In the recent past, the large language models (LLMs) [1]–[3] have achieved unprecedented success in Natural Language Understanding (NLU) [4] and Natural Language Generation (NLG) [5] based Natural Language Processing (NLP) [6] tasks such as text summarization, text classification, grammar correction, automated answering, text completion, etc. With prompt engineering, pre-trained LLMs such as GPT-3 [7], GPT-3.5 [8], Llama [9], etc. perform (NLP-based tasks) better than their predecessor transformer-based models such as BERT [10], RoBERTa [11], DistilBERT [12], SPANBERT [13], etc. The performance of automated question answering allows many downstream applications. Question answering systems are not only factually accurate but also can generate confident sounding, creative, and natural texts. On many occasions, it is very hard to distinguish between human written text and LLM generated synthetic text [14]–[16].

The advancement of synthetic text generation has raised some serious concerns, which have sociological and cyber effects. We can foresee that a large human workforce will be replaced by LLM-powered applications [17]. Chatbots are already performing many tasks in education, law, advertising, scientific/creative writing, entertainment, and many other industries. LLMs also empower cybercriminals with more widespread and dangerous capabilities [18]. Detecting the abuses of academic dishonesty, fake news/reviews, spam/phishing, etc., is more challenging [14], [19]. All educational institutes perceive the ChatGPT as a major challenge because all the traditional plagiarism-checking software are not trained to identify the AI-generated text. Therefore identifying a synthetically generated text will be a very challenging task in the future. Hence, in this research initiative, we plan to explore different methods to identify distinguishable features and patterns of synthetically generated text data.

AI-generated text, or in general machine-generated text, detection problem has become a very popular topic started from the Turing test and then used in chatbot evaluation [20]. In this paper, we focused on automated detection methods rather than human detection or hybrid methods. As summarized by Crothers, Japkowicz, and Viktor, automatic AI-generated text detection methods can be categorized into two general approaches: a) Feature-based and b) Neural Language models [14]. On the other hand, other research focused on the detection of specific domains such as academic settings [21]–[25], scientific [26]–[31], fake news/fake reviews/misinformation [32]–[36], etc.

In this work, we proposed two approaches to automatically detect AI-generated text: a) featured based with machine learning models and b) text similarity based methods. While our first approach follows the popular method in this domain, we improved the feature selection part and obtained higher detection results compared to existing methods. Later on, we verified our result with feature importance analysis and explainable models. In our second approach, given any text to be determined as AI-generated text, throughout the topic modeling and keyword extraction, we reverse the related questions and use AI models to generate the sample text. Based on the similarity analysis of the original text and the sample text, we can determine accurately whether it is AI-generated text or not. Our second approach works in general and in any specific domain without the requirement of ground truth data collection.

In summary, our contributions are listed below:

- A comprehensive study on handcrafted feature design and selection process with machine learning models can achieve an AI-generated text detection success rate of up

* Equal contribution

979-8-3503-0413-8/23/$31.00 ©2023 IEEE

to 0.9993.
- A novel machine text detection method based on reverse engineering and text similarity analysis.
- An analysis of feature importance and explanation models to discuss our results on AI-generated detection.

## II. Related Works

In this section, we summarized related research on AI-generated text detection methods.

### A. Feature-Based Approach

In this approach, different NLP techniques are used to extract useful features from the text. The motivation for this approach is from the observations that NLG models create different artifacts in generated text [14]. Some most important features that have been proposed are frequencies, linguistics, fluency, and fact verification [14], [37], [38]. These features are then trained with machine learning models, such as SVM, RF, or neural networks, to build classification models [39]–[43].

### B. Neural Language Model Approaches

For this approach, neural language models (NMLs) are built to detect generated text from state-of-the-art NLG models. Many NLG models, such as GPT or Grover, can be used to detect their own outputs. However, the performance is generally lower than a simple TF-IDF baseline model if no fine-tuning is performed. Therefore, fine-tuning pre-trained large language models is preferred in order to differentiate the human-written vs. NLG model output samples [14], [44], [45]. The disadvantages of this approach are a) the requirement for large datasets and b) heavy computation power and resources for training. This approach has been applied in detecting machine-generated text in a specific domain rather than in a general context.

### C. AI-Generated Text Detection on Specific Domains

Currently, there is no perfect AI-generated text detection in general (all possible domains). Much research focused on such detection tasks in specific domains, such as academic settings, scientific, fake news/reviews, or misinformation, as the scope can be narrowed down.

**Academic settings**: [21] collected submissions from computer science students and generated related ChatGPT responses. They evaluated 8 LLM text detectors on the above data. Their research has provided insights into the detectors' performance for the educator to maintain academic integrity. [22] conducted similar research but covered 12 publicly available AI-generated text tools and two commercial software (Turnitin and PlagiarismCheck). Similarly, other research, such as [23]–[25] contributed additional useful insights into the importance of AI-generated text detectors in academic settings.

**Scientific**: [26] studied the gap between the scientific content written by humans vs. generated by AI tools. The authors confirmed a "writing style" gap between them. [29] discussed the challenges of enforcing the policy on AI-generated papers/journals. [27] recently proposed a text representation method with machine learning models to detect fake scientific abstracts generated by a GPT-3 based model. Similarly, [28] studies the performance of using plagiarism detectors and blinded human reviewers on differencing original abstracts vs. fake abstracts generated by ChatGPT.

**Fake news/reviews or misinformation**: [32]–[34] studied the impacts of using AI-generated texts that can cause misinformation issues in the media. [36] conducted another study on the linguistic features and patterns of AI-generated COVID-19 misinformation. On the other hand, [35] proposed an AI model to detect AI-generated fake restaurant reviews on social media. These studies have raised the alarm about the severity of abused AI tools in media communications and information diffusion.

## III. Proposed Methodology

In this section, we discuss the proposed research methodology 1. The methodology comprises two main sections: data collection and experiments. We provide a description of each section here.

### A. Data Collection

To carry out our research, we require both synthetic and manually written text data for comparison purposes. As per our knowledge, there is no pre-existing database that provides this type of dataset. Therefore, our first step is to construct the dataset. In the second phase of our work, we will use NLP-based feature extraction to compare the features of the synthetic and human-generated data, train our model, and provide explanations.

We have generated two sets of data. The first one includes Wikipedia data and its corresponding artificially created data. The second one consists of news articles regarding the 2024 US election and their corresponding artificially created data.

*1) Dataset Based on Wikipedia:* We collected Wikipedia data using the method outlined in [46]. However, instead of only generating page summaries with ChatGPT, we expanded on this idea and created all sections of a Wikipedia page, which we labeled as synthetically generated data.

The process for generating synthetic text for a Wikipedia page about dogs is shown in Figure 2. The page contains different sections, including Taxonomy, Evolution, Biology, and more. The text from these sections is extracted and labeled as human-written text (HWT). To generate text related to Taxonomy, we prompted ChatGPT with "Describe Dog Taxonomy." The text generated by ChatGPT is labeled as synthetically generated text (SGT). The same process was used for all other sections of the Wikipedia page to generate both HWT and SGT.

*2) Dataset Based on News Articles:* We gathered a dataset on the upcoming US Election in 2024 by collecting news articles and generating synthetic data 3. Initially, over 500 URLs of news articles related to the election were collected. Next, text data is extracted from each article and identified important keywords. Using these keywords, ChatGPT generated five questions for each article and prompted ChatGPT to

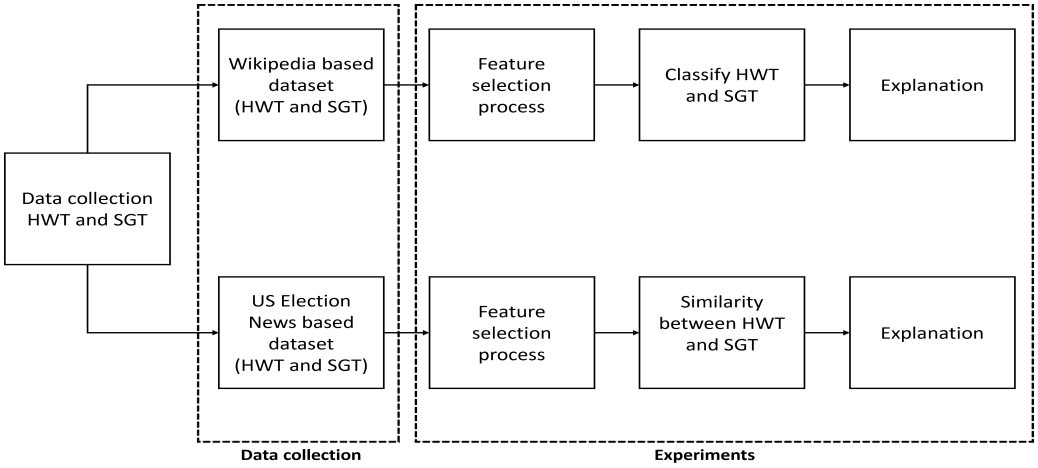



Fig. 1: Proposed Methodology

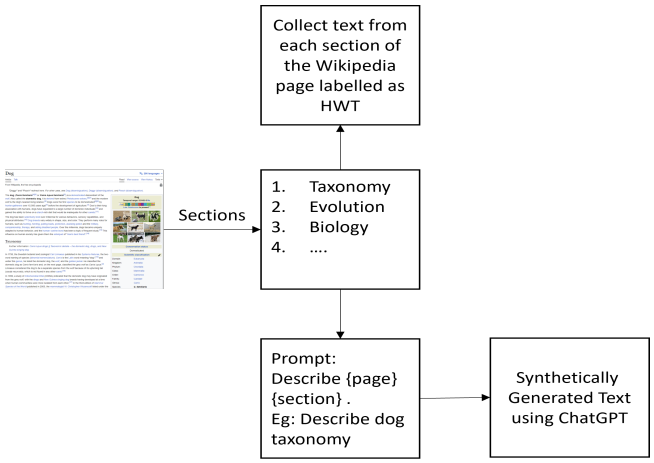



Fig. 2: Wikipedia based dataset generation process

answer those questions. As a result, we have generated five synthetic texts to correspond with each news article.

### B. Feature Engineering

When working with datasets for NLP tasks, it is crucial to carry out feature engineering. This involves using similar techniques for both datasets to gain a better understanding of their patterns and characteristics. These features will be useful in classifying the datasets later in this research. We have hand-crafted basic NLP-related features, TD-IDF [47] related features, N-gram features [48], topic modeling features, readability score, named entity recognition (NER) [49] count, and text error length features. To find an optimal set of features, principal component analysis (PCA) [50] is used.

### C. Classification

In order to differentiate between HWT and SGT, we utilized a dataset sourced from Wikipedia. We employed three different classifiers - Random Forest (RF) [51], Support Vector Machine (SVM) [52], and XGBoost (XGB) [53] - to perform the classification. Our models were trained, fine-tuned, and tested using this dataset. We used Precision, Recall, and F1 scores as metrics to evaluate and compare the performance of each model.

### D. Similarity Calculation

In order to assess the correlation between the HWT and SGT datasets pertaining to the 2024 US presidential election, the cosine similarity metric [54] has been employed to gauge their degree of similarity or dissimilarity. An overview of the similarity measure is offered for all the documents in the dataset.

### E. Explanations

The classification results come with a detailed explanation, which helps to identify the essential hand-crafted features. After evaluating the classifier models' performance, we compared and identified these features as common important ones for both model training and individual prediction.

## IV. Result and Discussion

In this section, data preprocessing and feature engineering will be described first. Then, the experiment setup, hyper-parameter tuning, and performance metrics are explained. Finally, the experiment results and discussion are presented.

### A. Dataset Preprocessing and Feature Engineering

*1) Data collection:* As explained in section III-A, we collected two different kinds of raw datasets: - Wikipedia based dataset: For each section in the original Wikipedia articles, we have the human-written text as the ground truth (extracted from the original article) and the corresponding ChatGPT-generated text. - US election 2024 related news article dataset: For each article related to the US election 2024 event, we have the human-written text (extracted from the article) and a maximum of 10 related ChatGPT generated text (in the form of question answering based on the extracted keywords from the article).

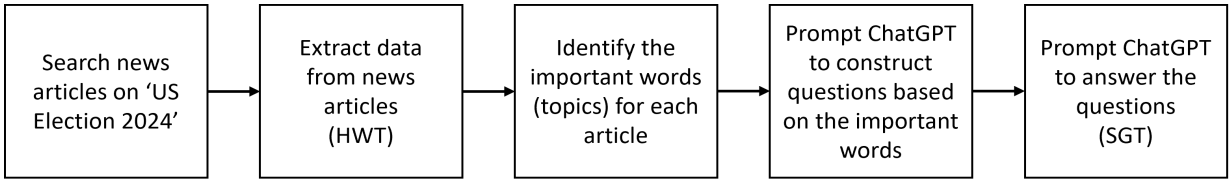



Fig. 3: US Election 2024 News Article based dataset generation process

In order to conduct the experiments, we need to preprocess the above two datasets and extract useful features from them, which will be described in the following two subsections.

*2) Data preprocessing:* For each text data in the above two datasets (either human-written or ChatGPT-generated ones), we conducted standard NLP preprocessing steps and implemented using NLTK [55], AutoCorrect [56], and BeatifulSoup libraries [57].

*3) Feature Engineering:* In order to perform NLP tasks with datasets, it's important to conduct feature engineering. This means using comparable methods for both datasets to gain a better understanding of their patterns and traits. These features will be helpful in later classifying the datasets during the research. We have manually created basic NLP-related features, TD-IDF-related features, N-gram features, topic modeling features, readability scores, named entity recognition (NER) counts, and text error length features. To determine the best set of features, we use principal component analysis (PCA) [50].

**Basic NLP features:**

- **Word_count**: The number of words in the given text
- **Word_density**: The average length of words in the given text. We calculate by taking the #characters / #words.
- **Punctuation_count**: The number of punctuations in the given text.
- **Title_word_count**: The number of title words (every first letter of the string is an upper case character) in the given text.
- **Upper_case_word_count**: The number of words starting with an upper case character.
- **Noun_count**: The number of noun lexicons.
- **Verb_count**: The number of verb lexicons.
- **Adj_count**: The number of adjective lexicons.
- **Adv_count**: The number of adverb lexicons.
- **Pron_count**: The number of pronoun lexicons.

**Term Frequencies and Ngram features:**

- **Count_vect**: The occurrences of words in the vocabulary.
- **Bigram_words**: The TFIDF features analyzed with the bigram model at the word level limiting to a maximum of 5000 features.
- **Trigram_words**: The TFIDF features analyzed with the trigram model at the word level limiting to a maximum of 5000 features.
- **BiTrigram_chars**: The TFIDF features analyzed with the bigram and trigram model at the character level limiting to a maximum of 5000 features.

TABLE I: Handcrafted feature descriptions and dimension size

| Feature Group | Descriptions | Feature Count |
|---|---|---|
| Basic NLP | char count, word count, word density, punctuation count, title word count, upper-case count, noun count, adv count, verb count, adj count, pro count | 11 |
| Term Frequencies and Ngram | Count_vect | 35742 |
| | Bigram_words | 5000 |
| | Trigram words | 5000 |
| | BiTrigram chars | 5000 |
| Topic modeling | NeuralLDA [58] | 20 |
| Others | readability score, NER count, text error length | 10 |

TABLE II: Dataset size

| | Wikipedia Dataset | US Election Dataset |
|---|---|---|
| HWT (0) | 3974 | 829 |
| SGT (1) | 4557 | 664 |
| Total | 8530 | 1493 |

**Topic Modeling features:** For each text data in the dataset, we analyzed the topic modeling features using the Neural LDA model [58] with the number of topics = 20.

**Other features:** Besides the above features, we analyzed the following additional important text features:

- **readbility_score**: 8 readability score of each text data (Flesch-Kincaid score [59], Flesch score [60], Gunning fog score [61], Coleman liau score [62], Dale Chall score [63], Ari score [64], Linsear write score [65], and Spache score [66]).
- **Named Entity Recognition (NER) count**: Analyzing and counting the number of NER tokens in the given text.
- **text_error_length**: The number of grammar errors in the text.

We collected a total of 50,783 features for each text data in our raw dataset. Most of the features are in the group of Term Frequencies and Ngram. Table I displayed the detailed list of feature descriptions and sizes of each component.

Table II displayed the size of our two datasets after preprocessing and feature extraction. We try to balance datasets with approximately equal numbers of data rows of both classes (human-written and ChatGPT generated texts).

### *B. Experimental Setup*

In this paper, we conducted three different kinds of experiments to demonstrate our proposed methodology in section III. In the first experiment, traditional machine learning algorithms were used to train classification models to identify

TABLE III: Hyperparameter settings and optimal values for the classification models

| ML models | Hyperparameter search range | Optimal |
|---|---|---|
| RF | n_estimators = {500, 1000, 1500, 2000} | 500 |
| | criterion = {gini, entropy} | gini |
| | min_samples_split = {2,4,6,8,10} | 10 |
| | min_samples_leaf = {2,4,6,8,10} | 10 |
| | max_features = {auto, sqrt, log2} | sqrt |
| SVM | C = {1,2,3,4} | 4 |
| | kernel = {linear, poly, rbf} | rbf |
| | degree = {3,5,7} | 7 |
| | gamma = {scale, auto} | scale |
| XGBoost | n_estimators = {500, 1000, 1500, 2000} | 1000 |
| | learning rate = {0.005, 0.01, 0.15} | 0.01 |

either human-written or ChatGPT generated text based on the collected handcrafted features. In this case, RF, SVM, and XGBoost were used to train our models. In the second experiment, we calculated the cosine similarity between the extracted features of human-written text vs. ChatGPT generated text and analyzed the results. Because a lot of term frequencies and n-gram features are sparse, we calculated the cosine similarity using three approaches: a) without term frequencies and n-gram features (WoTFNG), b) all features (AllFT), c) extracted PCA features from all features (PCA-FT). Finally, in our last experiment, we conducted the feature importance analysis to explain our models.

### C. Hyperparameter Tuning and Performance Metrics

In the first experiment, we used pandas for loading the extracted features and the scikit-learn [67] library to train the models and evaluate the performance. Grid searches were conducted to find the optimal parameters for the three machine learning models: RF, SVM, and XGBoost. The search ranges for the important hyperparameters are listed in Table III. For the RF model, the following other hyperparameters were used: min_weight_fraction_leaf = 0.0, min_impurity_decrease = 0.0, ccp_alpha = 0.0, and max_samples = None. For the SVM model, we used the following fixed hyperparameters: coef0 = 0.0, tol = 0.001, cache_size = 200, max_iter = -1, decision_function_shape = 'ovr'. In the second experiment, we used only one parameter for the PCA: n_components = 1024, which is the number of components that can be reduced from our feature set but can keep the maximum information.

To measure the performance of our classification models, we compared the following four important metrics: **accuracy, precision, recall,** and **F1 score**.

### D. Results and Discussions

*1) Classification Results:* Table IV displayed the overall performance of the machine learning models in our first experiment. While SVM could not perform well because of the complexity of the problem, both RF and XGBoost performed extremely well with an F1-score of 0.9993. This result showed that our proposed model with the suggested handcrafted features can help identify either human-written or ChatGPT-generated text. This supports our assumptions that there are differences in writing styles, the vocabulary used, text lengths, errors, and others when comparing them.

TABLE IV: Classifier Results for Wikipedia Dataset

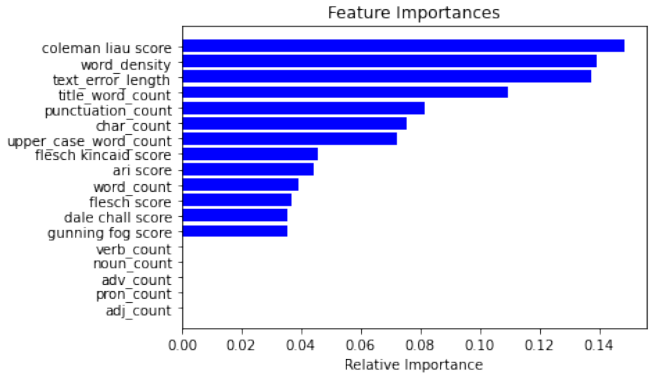
| | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| RF | 0.9993 | 0.9992 | 0.9993 | 0.9993 |
| SVM | 0.7421 | 0.7422 | 0.7389 | 0.765 |
| XGBoost | 0.9993 | 0.9993 | 0.9993 | 0.9993 |

*2) Document Similarity:* We utilized the Cosine similarity score to measure the similarity between HWT and SGT. The Cosine similarity score ranges from -1 to 1, where a score of 1 indicates that two documents are identical. If the score is less than 1, it means that the documents are not identical. As the score approaches -1, the degree of dissimilarity between the compared documents increases. In this research, we used the cosine similarity score to find the document similarity between HWT and SGT for the US Election related news articles based dataset. As mentioned in the III section, feature extraction is a prerequisite to calculating cosine similarity. Hence, we have calculated all features (as mentioned in Table I) before calculating the cosine similarity. Out of 50,783 features, most of them (35,742) belong to term frequencies, and they are sparse matrices. To observe the effect of the term frequencies, we have calculated two cosine similarities i) without term frequencies (15,041), and ii) with all features (50,783). In the first scenario, we are missing some information, and in the second scenario, we are using some features which are not needed. To solve both problems, we used PCA to identify the most useful features. Therefore, we used this new subset of features to find the cosine similarity score.

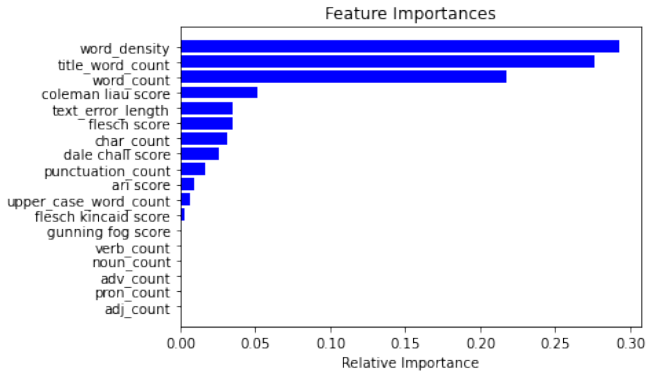
Table V shows the cosine similarity scores using different feature sets. For all three cases, 3643 HWT and corresponding SGT are used. The entire cosine similarity range is divided into four bins (-1 to -0.6, -0.6 to -0.2, -0.2 to 0.6, and 0.6 to 1), and a count of documents for each bin has been provided. Using the third feature subset, almost all document comparisons generate cosine scores between -0.6 to -0.2, which suggests these comparing documents are dissimilar. Using the first and

TABLE V: Cosine Similarity of US Election Dataset

| | **Features without TF-IFD** | |
|---|---|---|
| Cosine score | Document count | Document % |
| -1 to -0.6 | 2 | 0.05% |
| -0.6 to -0.2 | 2787 | 76.50% |
| -0.2 to 0.6 | 817 | 22.43% |
| 0.6 to 1 | 37 | 1.02% |
| | **All Features** | |
| Cosine score | Document count | Document % |
| -1 to -0.6 | 0 | 0.00% |
| -0.6 to -0.2 | 3537 | 97.09% |
| -0.2 to 0.6 | 106 | 2.91% |
| 0.6 to 1 | 0 | 0.00% |
| | **Features with PCA** | |
| Cosine score | Document count | Document % |
| -1 to -0.6 | 0 | 0.00% |
| -0.6 to -0.2 | 3642 | 99.97% |
| -0.2 to 0.6 | 1 | 0.03% |
| 0.6 to 1 | 0 | 0.00% |

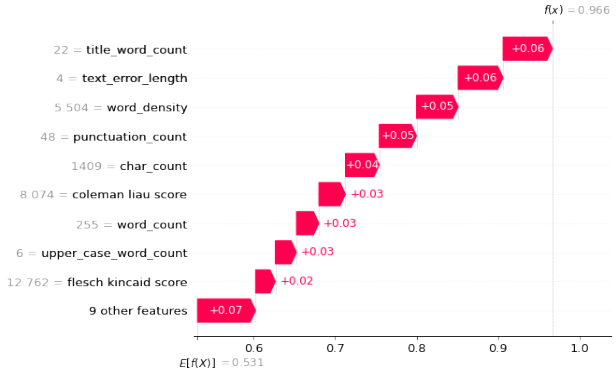



(a) Feature Importance for Random Forest Classifier

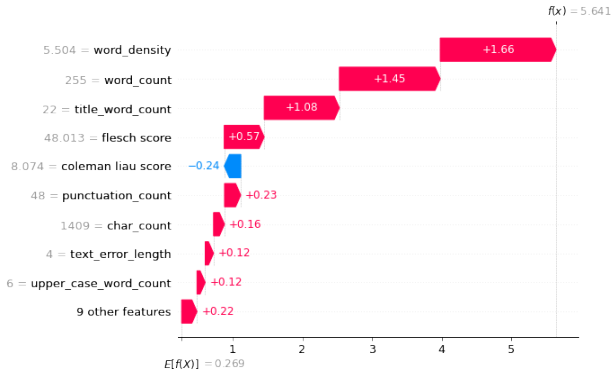



(b) Feature Importance for XGB Classifier

Fig. 4: Feature Importance Classifiers




(a) SHAP Explanation using Random Forest classifier




(b) SHAP Explanation using XGB classifier

Fig. 5: Feature explanation using SHAP for single text data point

second subsets of features, the cosine similarity scores suggest the comparing documents are similar. Hence the first two subsets of features are not performing well compared to the third set of features. The result also suggests that most of the documents have cosine similarity between -0.6 to -0.2, so setting -0.2 as the cosine similarity threshold value, we can classify the HWT and SGT with very high accuracy.

*3) Feature Importance Analysis:* For classification, the RF and XGB models performed similarly. To understand the behavior of the training features, we have calculated the feature importance of both models using scikit-learn library [68]. Figure 4 shows the relative training feature importance graph. We can see that for the RF classifier, Coleman liau score, word_density, text_error_length, title_word_count, and punctuation_count are the top five important training features; similarly, word_density, title_word_count, word_count, Coleman liau score, text_error_length are the top five important features for XGB classifier. Out of the top five features, four features are common for both models. This gives a fair idea about the set of important features that can be used as the distinguishing characteristics between HWT and SGT.

The Scikit-learn library has been used to calculate the importance of features in the entire training dataset. Additionally, the SHAP library [69] was utilized to explain the features for individual predictions. In figures 5a and 5b, the SHAP library has been used to display the individual predictions of the RF and XGB models in waterfall plots. These plots offer explanations for every prediction, utilizing the 10 most important features. Each row in the plot shows the positive (red) or negative (blue) contribution of each feature, moving from the expected model output over the background dataset to the prediction for that sample. The expected value of the model output is found at the bottom of the plot. SHAP explanations are given regarding the models' margin output before the logistic link function. The x-axis units are log-odds units, meaning that negative values imply a probability of less than 0.5 that the document is synthetically generated. The gray text before the feature names displays the value of each feature for that specific sample.

In Figure 5, the SHAP waterflow plot is displayed for both the RF and XGB models using the same text data. For the RF model shown in 5a, four of the top five SHAP features are also in the top five scikit-learn feature sets. It is worth noting that all four features have a positive impact on the decision-making process of the classifier. Similarly, for the XGB model, three out of the top five SHAP features are present in the top five scikit-learn feature sets, and all three features contribute to the decision-making process of the classifier. These findings suggest that handcrafted features are effectively contributing to the classification of the data points.

## V. Conclusion

In this paper, we proposed two approaches to detect AI-generated text: a) machine learning based and b) text similarity based methods. In order to test our proposed methodologies, we collected two different kinds of datasets: Wikipedia-based articles and US Election 2024 News articles. After that, we proposed four different groups of handcrafted features to be extracted from the raw text: Basic NLP features, Term Frequencies and Ngram features, Topic Modeling Features, and Other features (readability score, grammar error, and NER count). Then, we performed three experiments to demonstrate the effectiveness of our proposed methods.

In the first experiment, after extracting the handcrafted features, we trained three machine learning models using RF, SVM, and XGBoost on the classification of human-written or ChatGPT-generated text. Both RF and XGBoost models perform well, with an F1 score of 0.9993. This approach proves to be effective in any domain with high accuracy in detection. To use this approach, we need to collect the ground truth texts (human-written) and ask AI tools such as ChatGPT to generate the artificial text based on the forming related questions from the ground truth texts. Our designed handcrafted features can help develop effective machine-learning models for the detection of AI-generated texts.

The second experiment was designed to demonstrate our second approach to detecting AI-generated text. This approach does not require collecting ground truth texts. Given any text that we want to know, either human-written or AI-generated, based on topic modeling and keyword extraction, we formed related questions and asked AI tools to generate texts (ChatGPT in this project). Then, we applied a similar feature extraction in experiment 1. Finally, text similarity based on cosine will tell us the result. The results of our second experiment show that there is a clear classification boundary between human-written text and AI-generated text. This approach has proved effective and works in general situations, as no ground truth data needs to be collected.

To understand the roles and effectiveness of our designed handcrafted features, we conducted a feature importance analysis and built explainable models based on the SHAP library in our third experiment. According to our results, the Coleman liau score, word_density, text_error_length, title_word_count, word_count, and punctuation_count are the top important training features to help differentiate human-written or ChatGPT-generated texts.

## Acknowledgement

We thank Dr. Md Shohel Rana and Dr. Mohammad Nur Nobi for their valuable input and arguments on the project ideas and research methodologies.

## References

[1] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," 2007.

[2] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[3] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[4] N. Indurkhya and F. J. Damerau, *Handbook of Natural Language Processing*, 2nd ed. Chapman & Hall/CRC, 2010.

[5] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, ser. Natural Language Processing. Cambridge University Press, 2000. [Online]. Available: http://prp.contentdirections.com/mr/cupress.jsp/doi=10.2277/052102451X

[6] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009. [Online]. Available: http://www.nltk.org/book

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[8] "GPT-3.5 — lablab.ai," https://lablab.ai/tech/openai/gpt3-5, [Accessed 15-08-2023].

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019, cite arxiv:1907.11692. [Online]. Available: http://arxiv.org/abs/1907.11692

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[13] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the association for computational linguistics*, vol. 8, pp. 64–77, 2020.

[14] E. Crothers, N. Japkowicz, and H. L. Viktor, "Machine-generated text: A comprehensive survey of threat models and detection methods," *IEEE Access*, 2023.

[15] M. Jakesch, J. T. Hancock, and M. Naaman, "Human heuristics for ai-generated language are flawed," *Proceedings of the National Academy of Sciences*, vol. 120, no. 11, p. e2208839120, 2023.

[16] N. Köbis and L. D. Mossink, "Artificial intelligence versus maya angelou: Experimental evidence that people cannot differentiate ai-generated from human-written poetry," *Computers in human behavior*, vol. 114, p. 106553, 2021.

[17] "Will a Chatbot Write the Next 'Succession'? — nytimes.com," https://www.nytimes.com/2023/04/29/business/media/writers-guild-hollywood-ai-chatgpt.html, [Accessed 16-08-2023].

[18] M. Drolet, "Council Post: 10 Ways Cybercriminals Can Abuse Large Language Models — forbes.com," https://www.forbes.com/sites/forbestechcouncil/2023/06/30/10-ways-cybercriminals-can-abuse-large-language-models/?sh=74175f9304c0, [Accessed 16-08-2023].

[19] J. Pu, Z. Sarwar, S. M. Abdullah, A. Rehman, Y. Kim, P. Bhattacharya, M. Javed, and B. Viswanath, "Deepfake text detection: Limitations and opportunities," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1613–1630.

[20] D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck, "Automatic detection of generated text is easiest when humans are fooled," 2020.

[21] M. S. Orenstrakh, O. Karnalim, C. A. Suarez, and M. Liut, "Detecting llm-generated text in computing education: A comparative study for chatgpt cases," 2023.

[22] D. Weber-Wulff, A. Anohina-Naumeca, S. Bjelobaba, T. Foltýnek, J. Guerrero-Dib, O. Popoola, P. Šigut, and L. Waddington, "Testing of detection tools for ai-generated text," 2023.

[23] I. Dergaa, K. Chamari, P. Zmijewski, and H. B. Saad, "From human writing to artificial intelligence generated text: examining the prospects and potential threats of chatgpt in academic writing," *Biology of Sport*, vol. 40, no. 2, pp. 615–622, 2023.

[24] D. Yan, M. Fauss, J. Hao, and W. Cui, "Detection of ai-generated essays in writing assessment," *Psychological Testing and Assessment Modeling*, vol. 65, no. 2, pp. 125–144, 2023.

[25] E. Tulchinskii, K. Kuznetsov, L. Kushnareva, D. Cherniavskii, S. Barannikov, I. Piontkovskaya, S. Nikolenko, and E. Burnaev, "Intrinsic dimension estimation for robust detection of ai-generated texts," *arXiv preprint arXiv:2306.04723*, 2023.

[26] Y. Ma, J. Liu, and F. Yi, "Is this abstract generated by ai? a research for the gap between ai-generated scientific text and human-written scientific text," *arXiv preprint arXiv:2301.10416*, 2023.

[27] P. C. Theocharopoulos, P. Anagnostou, A. Tsoukala, S. V. Georgakopoulos, S. K. Tasoulis, and V. P. Plagianakos, "Detection of fake generated scientific abstracts," *arXiv preprint arXiv:2304.06148*, 2023.

[28] C. A. Gao, F. M. Howard, N. S. Markov, E. C. Dyer, S. Ramesh, Y. Luo, and A. T. Pearson, "Comparing scientific abstracts generated by chatgpt to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers," *BioRxiv*, pp. 2022–12, 2022.

[29] G. Hu, "Challenges for enforcing editorial policies on ai-generated papers," *Accountability in Research*, pp. 1–3, 2023.

[30] M. Perkins, J. Roe, D. Postma, J. McGaughran, and D. Hickerson, "Game of tones: Faculty detection of gpt-4 generated content in university assessments," 2023.

[31] Y. Liu, Z. Zhang, W. Zhang, S. Yue, X. Zhao, X. Cheng, Y. Zhang, and H. Hu, "Argugpt: evaluating, understanding and identifying argumentative essays generated by gpt models," 2023.

[32] S. Kreps, R. M. McCain, and M. Brundage, "All the news that's fit to fabricate: Ai-generated text as a tool of media misinformation," *Journal of experimental political science*, vol. 9, no. 1, pp. 104–117, 2022.

[33] A. Najee-Ullah, L. Landeros, Y. Balytskyi, and S.-Y. Chang, "Towards detection of ai-generated texts and misinformation," in *International Workshop on Socio-Technical Aspects in Security*. Springer, 2021, pp. 194–205.

[34] R. A. Partadiredja, C. E. Serrano, and D. Ljubenkov, "Ai or human: the socio-ethical implications of ai-generated media content," in *2020 13th CMI Conference on Cybersecurity and Privacy (CMI)-Digital Transformation-Potentials and Challenges (51275)*. IEEE, 2020, pp. 1–6.

[35] A. Gambetti and Q. Han, "Combat ai with ai: Counteract machine-generated fake restaurant reviews on social media," *arXiv preprint arXiv:2302.07731*, 2023.

[36] J. Zhou, Y. Zhang, Q. Luo, A. G. Parker, and M. De Choudhury, "Synthetic lies: Understanding ai-generated misinformation and evaluating algorithmic and human solutions," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3544548.3581318

[37] R. Tang, Y.-N. Chuang, and X. Hu, "The science of detecting llm-generated texts," 2023.

[38] G. Jawahar, M. Abdul-Mageed, and L. V. S. Lakshmanan, "Automatic detection of machine generated text: A critical survey," 2020.

[39] S. Chakraborty, A. S. Bedi, S. Zhu, B. An, D. Manocha, and F. Huang, "On the possibilities of ai-generated text detection," *arXiv preprint arXiv:2304.04736*, 2023.

[40] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi, "Can ai-generated text be reliably detected?" *arXiv preprint arXiv:2303.11156*, 2023.

[41] S. Mitrović, D. Andreoletti, and O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text," *arXiv preprint arXiv:2301.13852*, 2023.

[42] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, "Detectgpt: Zero-shot machine-generated text detection using probability curvature," *arXiv preprint arXiv:2301.11305*, 2023.

[43] N. Islam, D. Sutradhar, H. Noor, J. T. Raya, M. T. Maisha, and D. M. Farid, "Distinguishing human generated text from chatgpt generated text using machine learning," *arXiv preprint arXiv:2306.01761*, 2023.

[44] N. Lu, S. Liu, R. He, and K. Tang, "Large language models can be guided to evade ai-generated text detection," *arXiv preprint arXiv:2305.10847*, 2023.

[45] K. Hayawi, S. Shahriar, and S. S. Mathew, "The imitation game: Detecting human and ai-generated texts in the era of large language models," *arXiv preprint arXiv:2307.12166*, 2023.

[46] A. Bhat, "Introducing a Dataset to Detect GPT-Generated Text," https://towardsdatascience.com/introducing-a-dataset-to-detect-gpt-generated-text-96bb76dd2ed2, [Accessed 21-08-2023].

[47] J. Ramos, "Using tf-idf to determine word relevance in document queries," 1999.

[48] D. B. Paul, "Experience with a stack decoder-based hmm csr and back-off n-gram language models," in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

[49] A. Mikheev, M. Moens, and C. Grover, "Named entity recognition without gazetteers," in *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999, pp. 1–8.

[50] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[51] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[52] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[53] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[54] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *2016 4th International Conference on Cyber and IT Service Management*. IEEE, 2016, pp. 1–6.

[55] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[56] F. Sondej, "Autocorrect," https://github.com/filyp/autocorrect, [Accessed 21-8-2023].

[57] L. Richardson, "Beautiful soup documentation," *April*, 2007, [Accessed 21-08-2023].

[58] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," 2017.

[59] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," 1975.

[60] R. Flesch, "A new readability yardstick." *Journal of applied psychology*, vol. 32, no. 3, p. 221, 1948.

[61] W. H. DuBay, "Judges scold lawyers for bad writing," *Plain Language At Work Newsletter (Impact Information)(8)*, 2004.

[62] M. Coleman and T. L. Liau, "A computer readability formula designed for machine scoring." *Journal of Applied Psychology*, vol. 60, no. 2, p. 283, 1975.

[63] E. Dale and J. S. Chall, "A formula for predicting readability: Instructions," *Educational research bulletin*, pp. 37–54, 1948.

[64] R. Senter and E. A. Smith, "Automated readability index," Technical report, DTIC document, Tech. Rep., 1967.

[65] G. J. Christensen, "Readability helps the level," http://www.csun.edu/~vcecn006/read1.html, 2006, [Accessed 21-8-2023].

[66] A. J. Harris and M. D. Jacobson, "A comparison of the fry, spache, and harris-jacobson readability formulas for primary grades," *The Reading Teacher*, vol. 33, no. 8, pp. 920–924, 1980.

[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[68] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[69] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf