AI-Generated vs. Human Text: Introducing a New Dataset for Benchmarking and Analysis

Ali Al Bataineh, Rachel Sickler, Kerry Kurcz, and Kristen Pedersen

***Abstract*—Artificial Intelligence (AI) is increasingly embedded in our everyday lives. With the introduction of ChatGPT in November 2022 by OpenAI, people can now ask a bot to generate comprehensive writeups in seconds. This new transformative technology also introduces ethical, safety, and other general concerns. It is important to harness the power of AI to understand whether a body of text is generated by AI or whether it is organically human. In this paper, we create and curate a medium-sized dataset of 10,000 records containing both human and machine-generated text and utilize it to train a reliable model to accurately distinguish between the two. First, we use DistilGPT-2 with various inputs to generate machine text. Then, we acquire an equal sample size of human-generated text. All the text is cleaned, explored, and visualized using the Uniform Manifold Approximation and Projection (UMAP) dimensionality reduction technique. Finally, the text is transformed into vectors using several techniques, including BoW, TF-IDF, BERT, and neural network-based embeddings. Machine learning experiments are then performed with traditional models such as Logistic Regression, Random Forest, and XGBoost, as well as deep learning models like LSTM, CNN, and CNN-LSTM. Across all vectorization strategies and machine learning algorithms, we measure accuracy, precision, recall, and F1 scores. We also time each exercise. Each model completes its training within an hour, and we observe scores above 90%. We then use the SHapley Additive exPlanations (SHAP) package on machine learning models to explore if and how we can explain the model to further validate results. Lastly, we deploy our TF-IDF Random Forest model to a user-friendly web application using the Streamlit framework, allowing users without coding expertise to interact with the model.**

***Impact Statement*—Due to the rapid proliferation of AI-generated content, which poses significant challenges in distinguishing it from human-authored text, our paper provides the research community with a balanced dataset of 10,000 records that include both AI-generated and human-generated text. The dataset offers a standardized benchmark for training and evaluating AI models that detect AI-generated content and contributes to the ongoing discussions in AI ethics and the field of natural language processing.**

***Index Terms*—Artificial Intelligence, AI text detection, BERT, ChatGPT, DistilGPT-2, SHAP package, Explainable AI (XAI), OpenAI**

This work was supported by Norwich University Applied Research Institute and the AI Center at Norwich University, built under the grant "Advanced Computing through Experiential Education" from the Department of Education, award number: P116Z220106.

A. Al Bataineh is with the Artificial Intelligence Center, Norwich University, VT 05663, USA (e-mail: aalbatai@norwich.edu).

R. Sickler, K. Kurcz, and K. Pedersen are with the Norwich University Applied Research Institutes, VT 05663, USA.

Manuscript received Month 01, 2024; revised Month 02, 2024.

I. Introduction

A. Importance and Approaches to AI-Generated Text Detection

AI-generated text detection is an important and difficult task in the fast-growing landscape of Large Language Models (LLMs). With the increased proficiency of text generative AI, such as ChatGPT, many people cannot tell the difference between text written by a human or generated by AI. The approaches to detecting AI-generated text range from simple statistical methods to using generative models themselves for the task. The detection task is usually framed as binary classification. These methods include feature-based models such as Support Vector Machines (SVM), Random Forest (RF), and Neural Networks (NN), process input vectors with text-based features including Term Frequency-Inverse Document Frequency (TF-IDF), parts of speech, lemmas, punctuation count, and text length [1]. Statistical approaches like those using Markov methods can be also used as well as deep learning-based methods that leverage architectures like Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) for text analysis.

Generative adversarial networks (GAN) have been adapted to generate context for these models, and Reinforcement Learning (RL) has been used to help fine-tune the outputs of neural network models [1]. While the trained models have varying degrees of accuracy, a paper by Islam et al. demonstrated an extremely randomized trees model that outperformed other models with a 0.77 accuracy score when attempting to detect text that was generated by ChatGPT [2]. Still, a detection model created by Mitrovic´ et al. used DistilBERT (a transformer model) to do the same ChatGPT detection task with a 0.98 accuracy score. This paper also found that even these seemingly proficient models don't perform well when the text is rephrased using AI, with the detection accuracy falling to 0.79 [3]. Still, these models (and all models) only have the context they're provided and tend to only perform at even a moderately high level of accuracy on the output of the LLMs they were trained to detect. Using the generative models themselves seems like a promising way to detect the outputs of LLMs. Using a zero-shot method, the Grover model was used to demonstrate detection with 0.92 accuracy when limited to the domain on which the model was trained [4]. However, TF-IDF methods still outperform this method in general. State-of-the-art detection models are created by fine-tuning neural models with even a small sample of the data we seek to identify. This was demonstrated using RoBERTa-base (123M parameters) and RoBERTa-large (354M parameters)

using the machine-generated text created with GPT-2 using multiple sampling methods to increase generalizability [1]. This suggests that there is a sampling threshold that would allow a detection method to perform general AI-generated text detection. Not every detection method employs machine learning. For example, text readability and comprehensibility scores such as the Gunning-Fog Index and the Flesch Index show promise and have predictive application [1]. Vector databases are also used to assess similarity between prompt and response [5]. These provide a way for a human evaluator to make a determination as opposed to a strict binary classification problem resolved by a machine alone.

B. Challenges, Public Tools, and Future Directions

Some of the methods of detection have been deployed to publicly available web applications to help consumers distinguish between text generated by AI and that with a human author. None are 100% accurate and all have their limitations. GPTZero, which is trained specifically for ChatGPT, uses perplexity, a measure of text randomness, and burstiness, a measure of text complexity, as its methods of detection. Another tool, OpenAI Text Classifier, was created by OpenAI and claims to differentiate between human and AI-generated text and is trained on AI text generators including (but not limited to) ChatGPT. It has poor performance on short texts (less than 1000 characters), text generated from fine-tuned generative models, long text especially written by humans (identifies as written by AI), and text that is not in English. Some tools are very limited in scope, such as Writer.com's AI Content Detector which only allows a maximum of 1500 input characters. Copyleaks AI Content Detector touts a 99.12% detection accuracy rate and claims to work well with GPT-J, GPT-3, GPT-3.5, ChatGPT, GPT-4, and other related AI language models as well as multiple languages. Finally, the Giant Language Model Test Room (GLTR) compares the prediction it would make, by word, to the next actual word in a submitted text and allows the user to decide if the text in its entirety should be classified as AI-generated. When these tools were tested for accuracy with ChatGPT responses to seven prompts in an article published in the Journal of Applied Learning & Teaching [6], GLTR classified all sample texts as human. OpenAI Text Classifier and Writer.com's AI Content Detector classified only 2 of the AI-generated texts accurately while GPTZero classified three correctly. Copyleaks performed the best, classifying five of the seven texts as generated by AI. Even these tools which claim to have the highest accuracy on ChatGPT perform too inaccurately to be relied upon.

Unfortunately, the reliability of existing detection models plagues the AI-generated text detection field. The use of pseudorandom numbers in LLMs creates a distribution unlike the distribution of any text a human would create, and having even one line of AI-generated text within a human-authored text could throw a false positive [5]. In a paper by Liang et al., they find that over half of the 91 TOEFL (Test of English as a Foreign Language) essays they studied were identified as AI-generated by seven different AI detection systems, while the essays written by native English speakers were correctly classified [7]. This could lead to further marginalizing groups where authors may exhibit a more limited linguistic variability. Perhaps detection tools should not always be used as binary classification, but as feedback for the user to adjust or improve writing style because what these models are detecting is writing that is derivative of the training data.

There are ways to mitigate some of the issues with AI detection and appropriate use, for instance: rigorous testing for reliability and bias as well as tactics like using paraphrasers after using AI to generate text to evade detection. Many deep learning models and embedding methods, by their nature, do not offer explainability, which could be important to establishing authentic text and understanding the limits of classifiers that could be restricting discourse online, in academia, and beyond. Having a diverse training data set from many different generators (human and AI) could mitigate some of the distribution issues, making detection systems more reliable and simpler models could offer cleaner interpretability.

The training data used by researchers for creating and stress testing AI detection tools is as diverse as the models themselves. Some use publicly available datasets such as XSum [5] and the TweepFake Twitter dataset [1], while many also generated their own novel datasets manually, using a curated list of prompts to harvest AI-generated text from ChatGPT and BARD [3]. The human-authored text was often gathered from web scraping sites like CNN or restaurant reviews [2], Kaggle competitions [5], or from existing open-source datasets such as the "essays with instructions" dataset available on Hugging Face or the dataset provided by PoetryFoundation.org [8].

While most AI-generated text detectors are not better than a coin-flip in practice (and some are much worse), there are legitimate reasons someone might use an LLM that doesn't require the proverbial slap on the wrist of AI-detection. One could even argue that AI-assisted writing and translation lowers the barriers between people of different backgrounds and cultures to converse and share ideas. In addition to the text, features that could provide context to the text should be used to determine if messaging online is authentic or not.

Even if some uses of AI text generators are legitimate, we cannot ignore that AI-powered text generation is creating massive amounts of online data. In a 2023 paper, Shumailov et al. coined the term "model collapse" [9]. Model collapse is a degenerative process that results in a model that no longer represents the distribution of the underlying data. The probabilities of the AI-generated training data are compounded to create smaller probabilities for low-occurrence events and much higher probabilities for events that occur frequently. The authors call this the "curse of recursion" where much of the text found online could someday be generated by LLMs and thereby cause model collapse for future LLMs without a way to reliably detect authentic discourse from content that is model-generated. Knowing if the text was written by a human or not is imperative to its inclusion in LLM training data sets. LLMs can also be used for large-scale disenfranchisement of marginalized groups and exploiting systems and individuals for financial gain, among the many threats posed by public access. The ability to detect inauthentic discourse is essential

to the preservation of democracy. Academic fraud stressing the publication review process, fake resumes, and cover letters are already impacting the culture as well as overall reduced trust in text communications, whereby users assume opposing viewpoints are inauthentic rather than acknowledge the existence of a variety of viewpoints, further dehumanizing and polarizing societies. Distrust is fostered by open-source data collected that may be impacted by model-poisoning attacks where enough text data of a certain type is produced to impact the samples and therefore the distribution of model training data. Generated text can be used in mass online influence campaigns that can be political (propaganda), social (disinformation), as well as commercial (fake reviews). Phishing and spam can also fit into this category since AI-generated text content in this context is centered around targeting and scalability. Translation models can be used to distribute this information across language barriers and bots can be used to ensure it reaches the maximum audience while social algorithms can be exploited to amplify the message. The platforms where this type of generated text is most prevalent should use reliable and fair methods to detect and remove inauthentic content before it can cause harm. Even given all this, platforms need to consider the harm of mass-censoring online speech AI-generated text detection methods [1].

To further investigate the potential of AI text detection, this research makes the following key contributions:

- Introduce a balanced dataset of 10,000 records comprising both human- and AI-generated text.
- Conduct a comprehensive analysis using various vectorization techniques and machine learning models to achieve high accuracy in distinguishing between human and AI-generated text.
- Deploy our TF-IDF Random Forest model as a user-friendly web application using the Streamlit framework, allowing users without coding expertise to interact with the model. The code is openly shared on GitHub for further collaboration and improvement.

II. Methodology

In this section, we outline the methodology implemented in our research. Fig. 1 provides a visual representation of this process, beginning with data sourcing that includes both human-generated and machine-generated texts. It then shows the data cleansing process, followed by feature extraction. The final section of the figure outlines the modeling phases, where different machine learning algorithms were applied to the feature set for the development of the text classifier.

A. Data Collection and Preprocessing

To create our AI-generated text detection model, we first explored several means to acquire machine generated text and human generated text. Prerequisites included installing a virtual environment with the necessary packages as well as setting up the Kaggle CLI tool for downloading the human-generated text articles. We began with data acquisition and exploration, then model experimentation, research and development, followed by results, validation and explainability. A successful model is only as good as its data. It is important to have both human and machine generated text from diverse sources. We iteratively generated machine text by feeding various inputs to a Hugging Face DistilGPT2 model using a pipeline from the transformer package. During the development phase, we realized many of the outputs were similar unless the inputs were also very unique. Meanwhile, each execution of the DistilGPT2 pipeline used a single input and produced a single output. As we wanted thousands of diverse outputs, it was important to automate an iterative process using unique inputs. Thus, to minimize duplication, we used a unique input per output. DistilGPT2 is an English-language model pre-trained with the supervision of 124 million parameter version of GPT-2, an OpenAI product. DistilGPT2 has 82 million parameters and was developed using knowledge distillation, designed to be a faster, lighter version of GPT-2 [10]. On average, DistilGPT2 is two times faster than GPT-2. The distillation transformer has also been applied to various flavors of BERT such as RoBERTa to create DistilRoBERTa, German BERT to create German DistilBERT, and Multilingual BERT to create DistilmBERT which supports 104 languages [11].

Although GPT-4 was released in March 2023, our study utilized DistilGPT-2, chosen for its availability as a free API on Hugging Face. Considering the advancements in AI, future research could investigate the integration of more recent models like GPT-4 to potentially enhance performance. In our current setup, producing a single record took a few seconds, as each input required a separate query to the API over the Internet connection. In future iterations of this research, we aim to improve the efficiency of data collection, possibly by enabling the retrieval of multiple records per API request.

The DistilGPT2 transformer takes any text prompt on which to concatenate more words using machine learning. The transformer also accepts parameters such as max_length, pad_token_id, and num_return_sequences. We attempted to extract the longest length by iterating through a high volume of a randomized selection of prompts. Using the highest value discovered to be possible (max _length=512), a pad _token _id of 50256 (for open-ended text generation) and 2 num _return_sequences, we selected the longer of the 2 produced output texts.

Two key datasets were used as inputs for the machine generated text model. This includes WMT16, a dataset typically used for translation. This dataset was desirable for this use case because it is a collection of key-value pairs, where the key is an English excerpt, and the value is a translated equivalent. WMT16 contains several languages. We extracted only the English excerpts, resulting in a long, diverse list of short English text blobs. The second dataset is English Rotten Tomatoes movie reviews, which are of a similar length as the WMT16 excerpts. Both datasets were loaded using the Hugging Face datasets package.

With a resulting 4665 records with Rotten Tomato inputs and 4533 machine-generated records with WMT16 inputs, the final pandas dataframe is modified to include 4 columns: one for the machine-generated text, one for the label (1 for machine), the input source (Rotten Tomato or WMT16), and the text word count. The resulting 9266 records with an

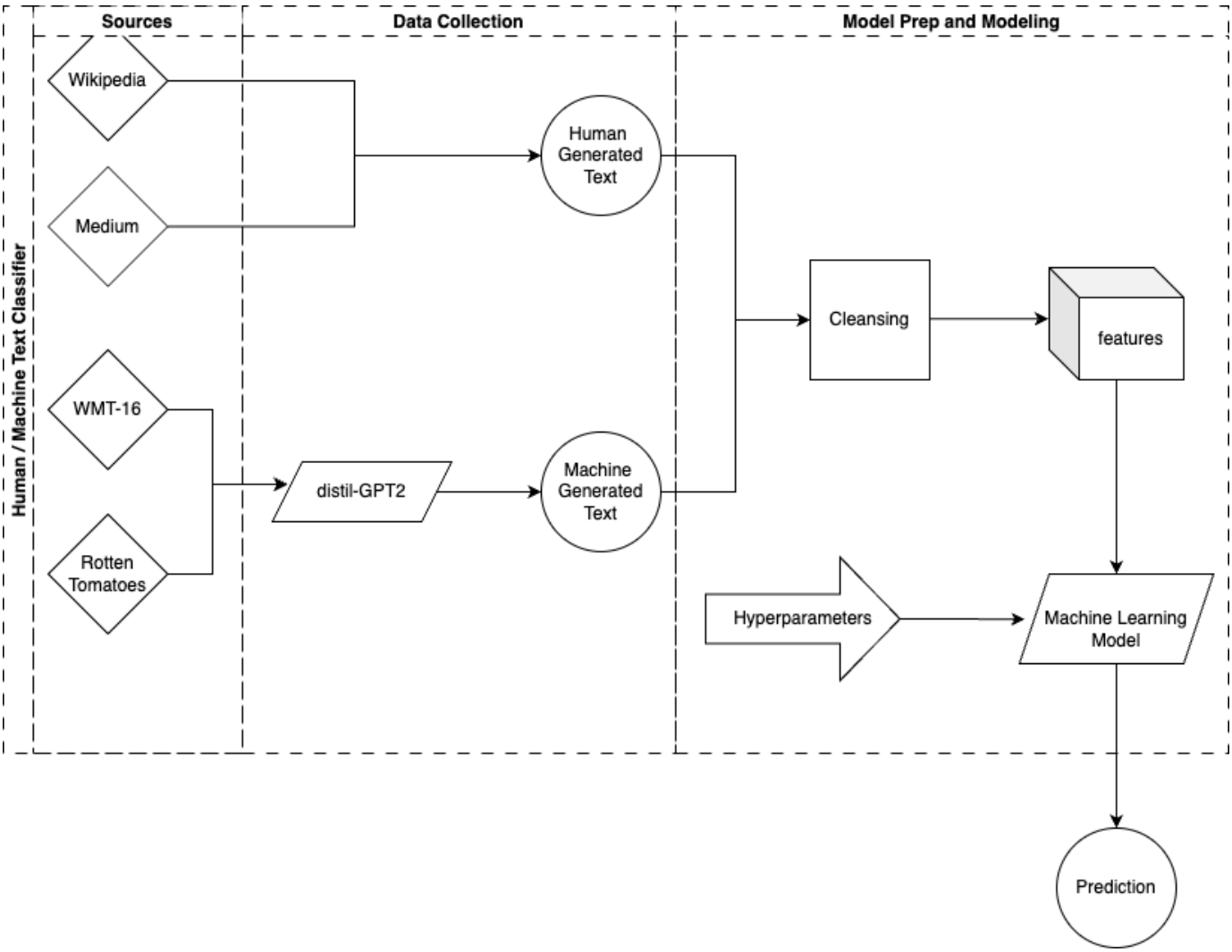


Fig. 1: Overview of the Methodological Process.

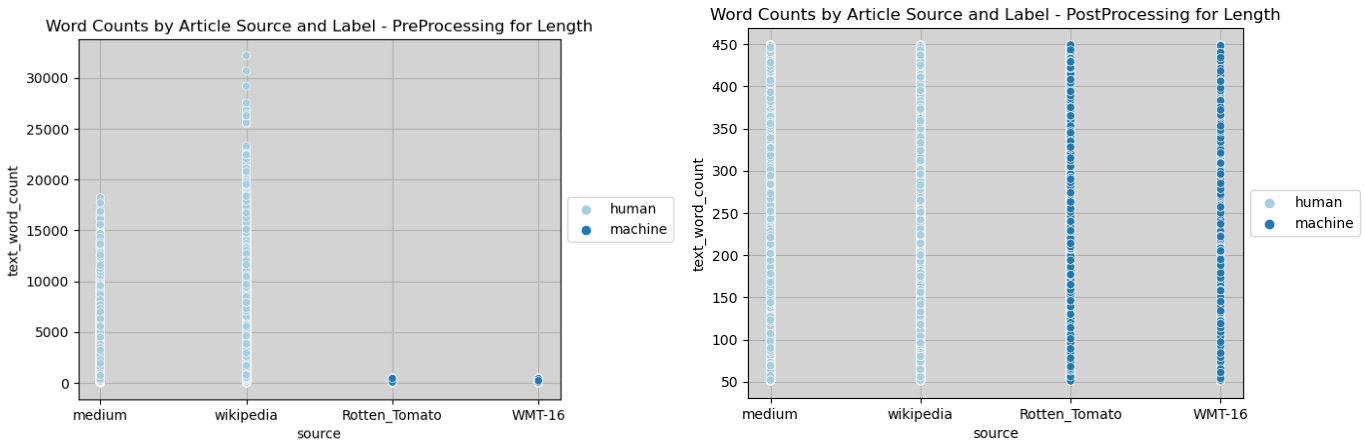
average word count of 214 are saved as a csv.

Collecting human-generated text was much simpler using the Kaggle CLI tool for two different sources. First, over 185,000 Medium articles were downloaded with an average word count of 834. Second, 19,990 Wikipedia articles were downloaded using the load_dataset module of the datasets package by Hugging Face. Wikipedia had an average word count of 3236. All of the Medium and Wikipedia articles were labeled as a 0 for human and saved as a csv. The resulting 200,000 or so human-generated records needed cleaning. To begin, the word counts were significantly higher than the machine generated text. To prevent word count bias causing the machine learning algorithms to predict human versus machine based on article length alone, we cleaned up the data among word counts.

In Figs 2(a) and 2(b), the machine generated sources Rotten Tomato and WMT16 are labeled with a dark blue, with light blue representing human generated sources Medium and Wikipedia. On the left in Fig. 1a, each point represents an article. The y-axis illustrates word counts. It is evident in Fig. 2(a) that the human generated sources have many more articles with larger word counts than the machine generated sources. Fig. 2(b) represents the dataset after cleansing both the human and machine generated sources of any article with a word count exceeding 450 or under 50, keeping 5000 records of each type, without duplication. After processing, the average machine generated word count is 276 and average human generated word count is 256.

We also visualized the dataset by label (machine or human, 1 and 0, respectively) and by source. This is shown in Fig. 3. This balanced distribution among source and word counts is much better for machine learning consumption, albeit the record count decreases to 10,000. In future work, we may increase the dataset size. However, this also permitted much faster processing times. Preprocessing steps also include data cleansing for punctuation, non-ascii characters, empty spaces, tabs, and end of the line characters. Exploratory data analysis also included part-of-speech tagging for verbs, adjectives, adverbs, and nouns. The resulting vocab is 82,292 unique terms. This vocab is used to control the size of the CountVectorizer and TFIDFVectorizer matrix.

In our commitment to transparency and reproducibility, the



(a) Word Counts PreProcessing

(b) Word Counts PostProcessing

Fig. 2: Comparison of Word Counts Before and After Processing

dataset utilized in this study is publicly available for review and further research at https://github.com/rsickle1/human-v-ai.

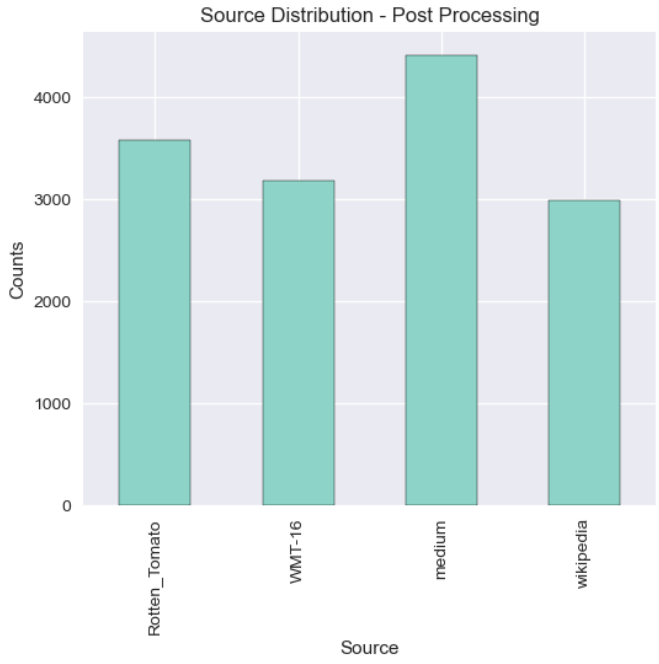


Fig. 3: Data Labels by Source

B. Model Experimentation, Research, and Development

To optimize the balance between processing time and accuracy scores, several vectorization techniques were explored. These include BoW, TF-IDF, BERT, and neural network embeddings with the keras package. BoW refers to a simple word frequency count. TF-IDF considers the term position in the document. BERT is calculated using the transformer mechanism and is able to represent text with full context.

The nature of word vectors is that they are highly dimensional to represent the text in multiple ways. To visualize highly dimensional data, dimensionality reduction techniques can be applied for representing two or three dimensions which can be more easily represented on a graph. We applied the UMAP algorithm to view BoW vectors by label and then by source in Figs 4 (a) and 4 (b), respectively.

As the below representations are essentially clusters of the 2-dimensional data, we would ideally see integrated colors. This would mean that the data does not follow any obvious pattern; the data is random. We don't see this in the BoW graphs, however. We have better success in the TF-IDF two-dimensional graphs in Figs 5 (a) and 5 (b). although it still appears as if the source of the dataset tends to lean one way or another. We see this in Fig 5. b in the heavy cluster of red representing the Rotten Tomato dataset, Wikipedia in purple tends to lean lower left, and medium articles in green and WMT16 in orange are somewhere in the middle. The BERT embeddings in Figs 6 (a) and 6 (b) appear much more spread out than the BoW and TF-IDF vectors, however, it still seems as if TF-IDF manages to integrate best of the three representations.

The BoW vectors were calculated using the CountVectorizer module from the sklearn package [12] using an ngram range of (1,2), a list of English stop words provided by the nltk package, and a provided vocabulary list of about 82,292 terms which was derived during the cleansing preprocessing steps. Other parameters were explored but ultimately abandoned for being too memory intensive either to the point of crashing the local machine or without adding value to the model.

The TF-IDF vectors were calculated using the same n-gram range, the same stop words, but without the vocabulary list and instead controlling size using the min_df parameter with a value of 10. This parameter removes terms that appear too infrequently. Using a value of 10 ensures that any term used in the TF-IDF matrix has appeared in at least 10 documents (or records).

The BERT embeddings were calculated using the SentenceTransformer package, where msmarco-distilbert-dot-v5, multi-qa-distilbert-cos-v1, and all-distilroberta-v1 were explored having a max sequence length of 512 and using 768 dimensions (each word is represented as a vector with a

length of 768). As another incentive to trim down the dataset, should the text exceed 450-word counts, this package would truncate any term count greater than 512. BERT embeddings are computationally expensive to calculate and took over an hour to compute on a Google Colab machine. Because of this, results were serialized using the pickle package and imported for future use, only being re-calculated if the underlying dataset for some reason changed (such as new cleansing techniques).

From here, we feed each vectorization strategy to different models. These include Logistic Regression, Random Forest, and XGBoost. Logistic Regression [13] uses the sigmoid function to predict the probability of an outcome. in this case categorizing text as either machine-generated or human-generated. Random Forest [14] employs a series of Decision Trees to produce an outcome. XGBoost, or Extreme Gradient Boosting [15] creates an ensemble of weak decision trees sequentially. Each tree in XGBoost corrects the errors of its predecessors, enhancing the overall predictive accuracy.

III. Results and Discussion

A. Evaluation of Machine Learning Algorithms

In the first part of our study, we evaluated the performance of three traditional machine learning algorithms - Logistic Regression, Random Forest, and XGBoost - across three different text vector representations: Bag of Words (BoW) [16], Term Frequency-Inverse Document Frequency (TF-IDF) [17], and Bidirectional Encoder Representations from Transformers (BERT) [18]. This comprehensive analysis aims to determine how each algorithm performs with different representations and to identify the most effective combination for text classification tasks. Different parameters were explored for these models. Initially, we experimented with different cross-validation (CV) values [19]. Interestingly, a CV value of 3 yielded results comparable to those obtained with higher CV values, but with reduced computational time. Thus, we continued using three CV with the remaining models.

Table 1 presents the results for the BoW vector representation. The XGBoost classifier outperforms the others, delivering uniformly high scores of 0.91 across accuracy, precision, recall, and F1 metrics. This impressive performance underscores XGBoost's ability to effectively navigate the complexities of the BoW feature space. Its robust ensemble approach adeptly manages bias and variance within the dataset. Following closely is Logistic Regression, which demonstrates its effectiveness with balanced performance metrics, all approximately around the 0.90 mark.

The results for the TF-IDF representation are presented in Table 2. Both the Random Forest and XGBoost classifiers achieved identical scores across all metrics—accuracy, precision, recall, and F1 score—with each scoring 0.91. This outcome underscores the effectiveness of these algorithms in utilizing the TF-IDF representation for classification tasks. Logistic Regression, while marginally lower, still demonstrated commendable performance, maintaining scores of 0.90 across all metrics.

The evaluation of the BERT representation in Table 3 revealed a slightly divergent trend. While Logistic Regression sustained its performance, achieving a 0.90 score across all metrics, the Random Forest and XGBoost classifiers experienced a diminution in performance, with scores descending to 0.81 and 0.83 respectively. This decrement may be attributed to the sophisticated nature of BERT embeddings, which could necessitate more complex model architectures or fine-tuning to fully capitalize on the contextual cues within the data. Logistic Regression's consistency across all three representations substantiates its versatility and underlines its capability to deliver balanced classification outcomes regardless of the vectorization technique employed. On the other hand, the decline in performance of Random Forest and XGBoost when transitioning to BERT embeddings merits further investigation. This could potentially suggest that while these algorithms excel with traditional vector representations, they may not be as adept at harnessing the full potential of contextual embeddings without significant adjustments to their configuration.

TABLE I: Classification performance of different models on BoW representation

Classifier Type	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.90	0.90	0.90	0.90
Random Forest	0.87	0.88	0.87	0.87
XGBoost	0.91	0.91	0.91	0.91

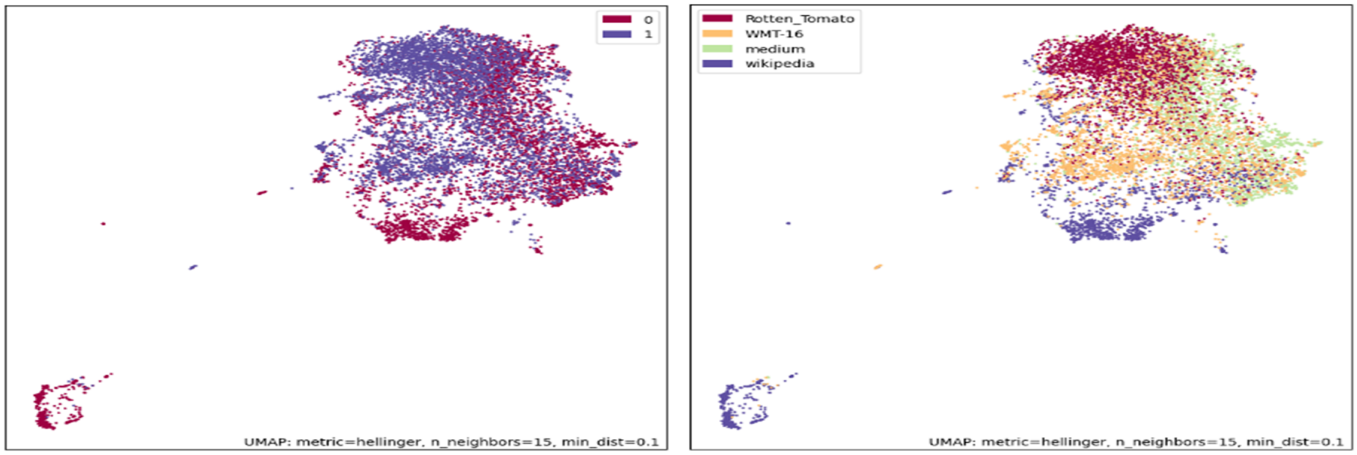
TABLE II: Classification performance of different models on TF-IDF representation

Classifier Type	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.90	0.90	0.90	0.90
Random Forest	0.91	0.91	0.91	0.91
XGBoost	0.91	0.91	0.91	0.91

TABLE III: Classification performance of different models on BERT representation

Classifier Type	Accuracy	Precision	Recall	F1 Score
Logistic	0.90	0.90	0.90	0.90
Random Forest	0.81	0.81	0.81	0.81
XGBoost	0.83	0.83	0.83	0.83

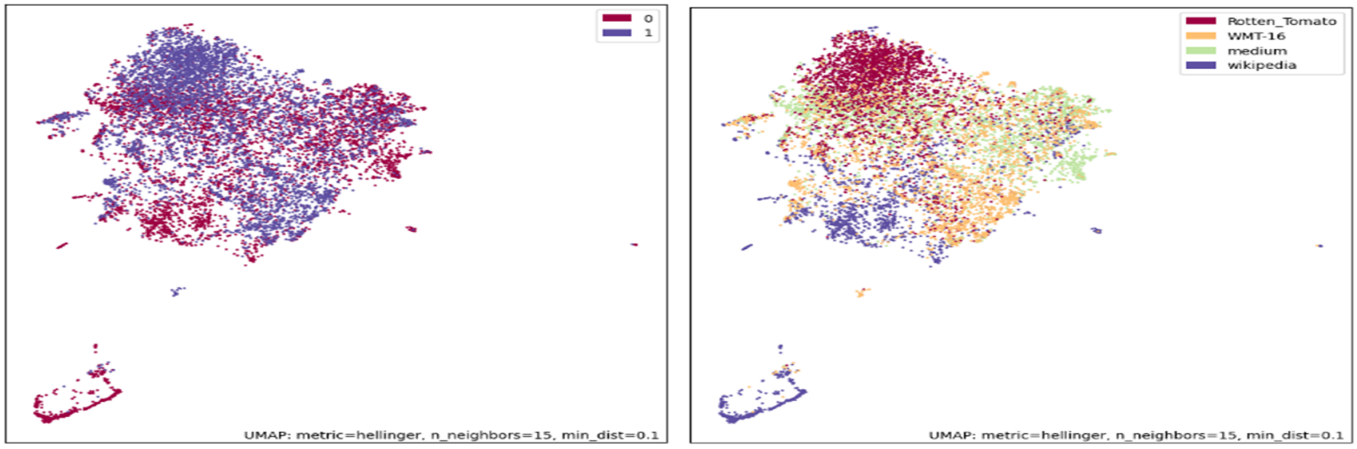
To further our understanding of the decision-making processes underlying our machine learning models, we employed SHAP (SHapley Additive exPlanations) analysis [20]. We used the XGBoost model to most easily integrate with the SHAP package's Tree Explainer to produce the below Summary Plots per vectorization strategy. The X data is the Count Matrix or TF-IDF Matrix or in the case of BERT, the BERT embeddings. A dataframe of the X training data is sampled randomly for 100 records, upon which SHAP values are calculated. This can be an expensive process, hence the random sampling of 100. The summary plot takes the parameters SHAP values and the 100 randomly selected X training records. The results are given in Fig. 7. The purpose of these plots is to demonstrate based on SHAP values what is likely causing the positive, negative, or neutral impact on the model output. SHAP values



(a) Distribution by Label — depicting the clustering of human-generated (0) and machine-generated (1) text

(b) Distribution by Source — showing the separation of texts from Rotten Tomato, WMT-16, Medium, and Wikipedia

Fig. 4: UMAP Visualizations of Text Data Vectorized by BoW, Categorized by Label and Source.



(a) Distribution by Label — depicting the clustering of human-generated (0) and machine-generated (1) text

(b) Distribution by Source — showing the separation of texts from Rotten Tomato, WMT-16, Medium, and Wikipedia

Fig. 5: UMAP Visualizations of Text Data Vectorized by TF-IDF, Categorized by Label and Source

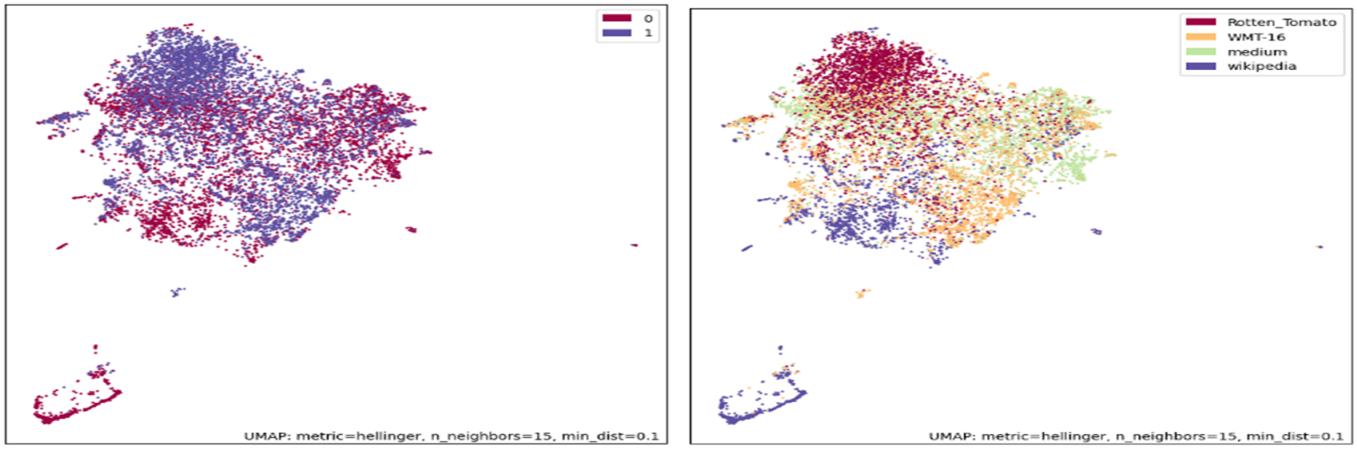
range from –4, to 0, to 4. A value of 0 for some feature means that in that observation, the value of that feature is ignored by the model; the model is neutral to this feature. SHAP values of 4 are strongly positive; the value of that feature in the current example increases the model output by 4.

For example, in the BoW and TF-IDF models, the term nt is often associated with a predicament in which the model makes a positive prediction (machine-generated text). Meanwhile, the term reference often causes the model to make a negative prediction (human-generated text). The term link is consistently more valuable to the model than the term let.

However, for the BERT SHAP model, we can see much more activity per term, but the terms are ambiguous. Indeed, the most important takeaway from this Figure is that BERT embeddings cannot be explained simply. In regulatory settings with requests to Right to Explanation, transformer-based Artificial Intelligence is much more complicated to provide straightforward explanation.

B. Exploration of Deep Learning Techniques

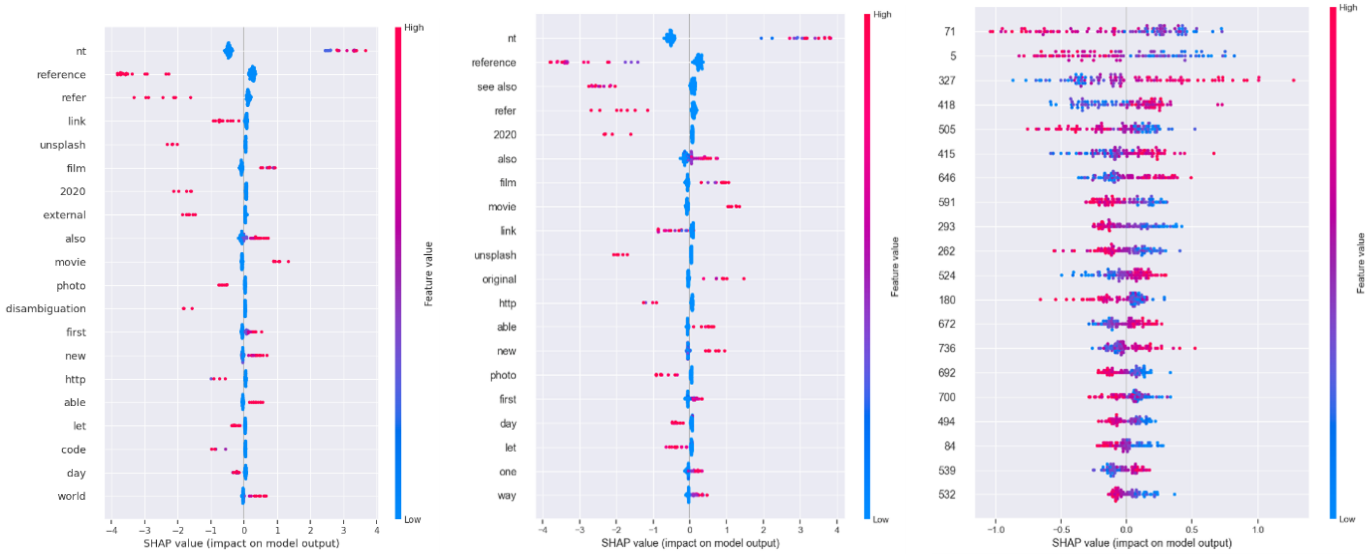
The latter phase of our study concentrated on evaluating deep learning strategies for AI-generated text detection, utilizing three distinct neural network architectures: LSTM [21], CNN [22], and a hybrid CNN-LSTM model [23]. These models were implemented using the Keras library [24]. our dataset underwent tokenization via 'Keras' Tokenizer, subsequently forming the basis for model input. Each model was constructed with an Embedding layer designed to handle a vocabulary size of 10,000 words, using an embedding dimension of 16 to represent these words densely, and processing sequences with a maximum length of 500 tokens. This setup was consistent across the LSTM, CNN, and CNN-LSTM models, ensuring a uniform foundation for feature representation. The LSTM network was structured sequentially, incorporating an Embedding layer, an LSTM layer with 32 units, and a Dense layer with a sigmoid activation function for binary classification. Our CNN architecture replaced the LSTM layer with a Conv1D layer, also with 32 units, and included a



(a) Distribution by Label — depicting the clustering of human-generated (0) and machine-generated (1) text

(b) Distribution by Source — showing the separation of texts from Rotten Tomato, WMT-16, Medium, and Wikipedia

Fig. 6: UMAP Visualizations of Text Data Vectorized by BERT, Categorized by Label and Source.



(a) BoW SHAP Summary

(b) TF-IDF SHAP Summary

(c) BERT SHAP Summary

Fig. 7: SHAP Summary Plots for BoW (left), TF-IDF (center), and BERT (right) vectorization strategies.

Flatten layer to process the spatial information. The CNN-LSTM combined the two approaches, utilizing both Conv1D and LSTM layers with 32 units each, aiming to exploit the advantages of convolutional feature extraction and sequence processing. To prevent overfitting and optimize computational efficiency, early stopping was employed during the training phase. The effectiveness of this approach is illustrated in Fig. 8 which provides the training and validation accuracy and loss curves for each of the deep learning models. These curves are pivotal for understanding the models' learning behavior over epochs and for identifying potential issues such as overfitting or underfitting. We summarize the hyperparameters and the final results obtained at the last epoch before early stopping in Table IV and Table V. The LSTM model demonstrated an accuracy of 98.75% on the training data and 88.90% on the validation data, with a notable loss of 0.3532 on the validation set after four epochs. The CNN model, on the other hand, achieved a slightly higher accuracy of 98.90% on the training data and 90.80% on the validation data, but with a higher validation loss of 0.3950 after three epochs. The CNN-LSTM model's performance was slightly lower, with an accuracy of 98.76% on the training data and 87.90% on the validation data, resulting in a validation loss of 0.3986 after four epochs. Notably, We found that the CNN model trained fastest and provided the best results, making it a primary candidate for further research and optimization.

C. Deployment to a Web Application

The underlying mechanism of the application involves de-serializing a pre-trained sklearn pipeline object, enabling the input text—such as the example shown in Fig. 10, captured from an AI-related Wikipedia page—to be processed and classified. The application translates the binary classification result into a user-friendly output displayed on the UI.

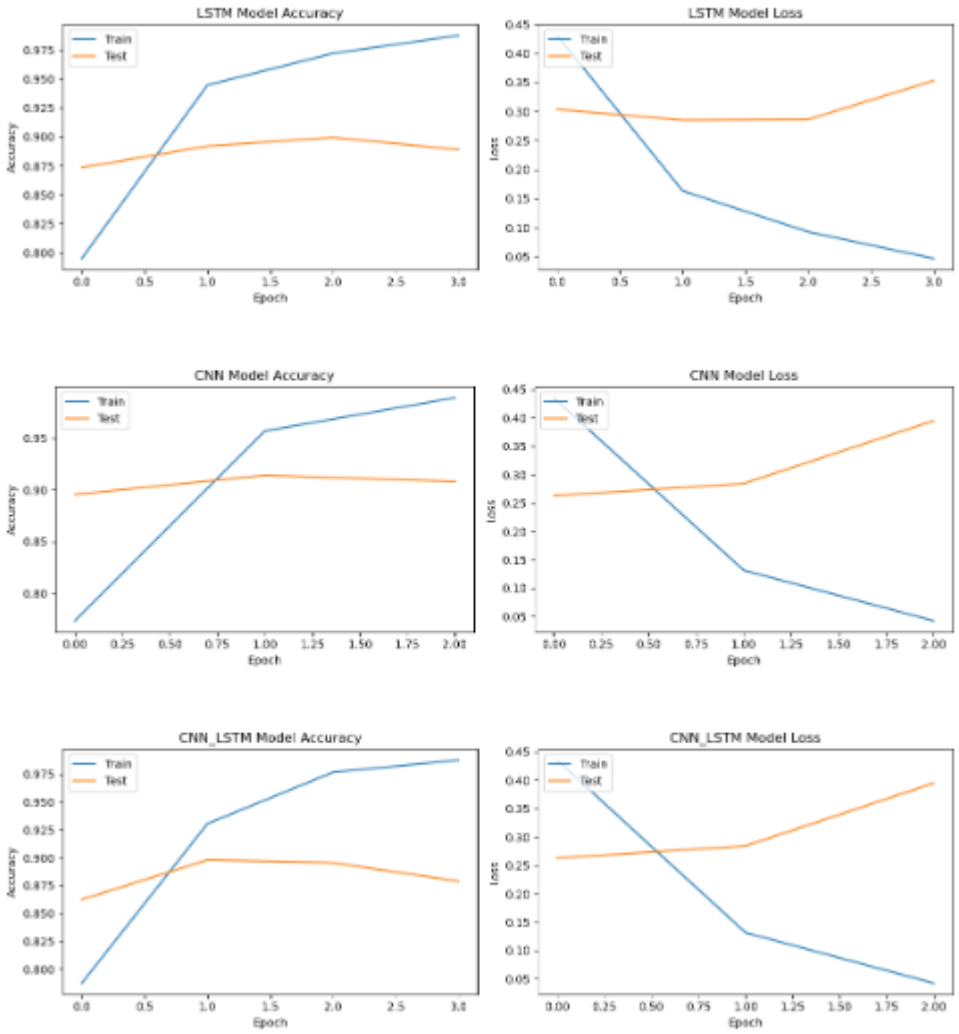


Fig. 8: Training and validation accuracy and loss for LSTM, CNN, and CNN-LSTM models.

TABLE IV: Hyperparameters for Deep Learning Models

Model	Max Features	Maxlen	Embedding Dim	Units	Training Time (s)
LSTM	10000	500	16	LSTM: 32	107.31
CNN	10000	500	16	Conv: 32	9.21
CNN-LSTM	10000	500	16	Conv: 32, LSTM: 32	127.19

In an effort to bridge the gap between technical research and practical utility, We deployed the TF-IDF random forest model from the experiments above to a web application with a user-friendly user interface (UI). We created this application using the Streamlit framework (https://streamlit.io/) which provides components to allow the developer to create a frontend application using pure Python. This removes the barriers to most machine learning and data science professionals that would prevent them from providing their models to users who don't write Python code. This is also a way to allow users outside of the coding community to test models and provide feedback to the technical resources responsible for training the models. The application could also be deployed to a web server to make it publicly available. The GitHub repository for this application can be accessed here: https://github.com/rsickle1/human-v-ai

The user interface, depicted in Figure 9, showcases the simplicity and intuitiveness of the application, allowing users to input text and receive model predictions with ease.

This streamlined interface ensures that the model is not only accessible to those outside the technical domain but also provides a platform for extensive user feedback. Additionally, the application is structured to be deployed on a web server, thereby enhancing its accessibility to a wider audience.

The flexibility of the application's architecture is such that any serialized model, equipped with the requisite preprocessing steps and capable of binary inference, can be integrated into the system. This facilitates a collaborative environment where machine learning experts can deploy their models and garner feedback from a diverse set of users, including subject matter experts. Such a practice is instrumental in refining and

TABLE V: Final Training and Validation Results for Deep Learning Models

Model	Train Loss	Train Accuracy	Val Loss	Val Accuracy	Epochs
LSTM	0.0467	98.75%	0.3532	88.90%	4
CNN	0.0423	98.90%	0.3950	90.80%	3
CNN-LSTM	0.0447	98.76%	0.3986	87.90%	4

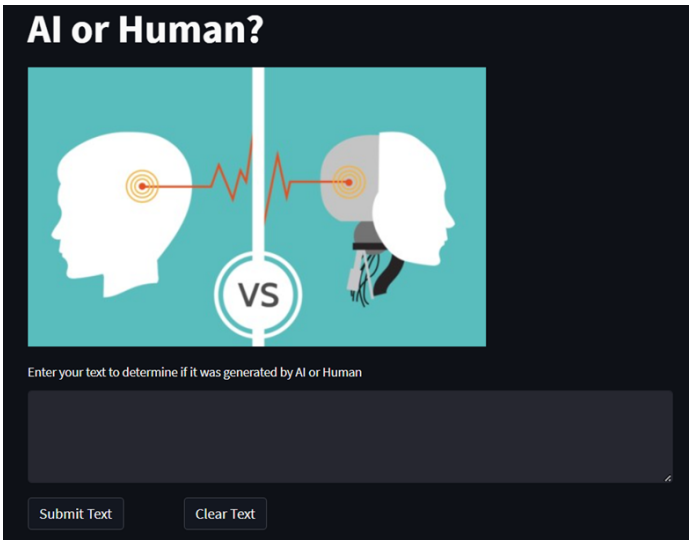


Fig. 9: Screenshot of the Streamlit web application user interface.

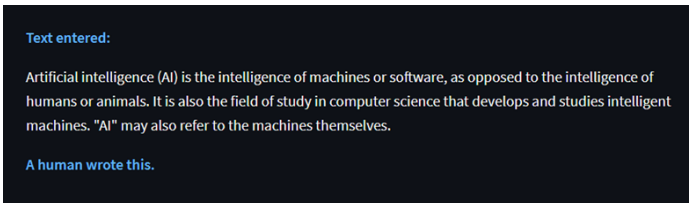


Fig. 10: Example output displayed by the web application, using text from the AI Wikipedia page as input.

advancing AI-detection systems, making them more robust and reliable.

D. Limitations of the Study

This research, while comprehensive in its approach and methodologies, encounters several significant limitations that warrant consideration.

A primary limitation lies in the inherent ”black box” nature of the deep learning models utilized in the study. These models, particularly the complex architectures like CNNs and LSTMs, often operate without transparent interpretability. Despite their proven effectiveness in text classification tasks, the lack of clarity in how these models process inputs and make predictions poses a considerable challenge. This aspect is particularly crucial in fields where the need for explainability and accountability is paramount. The inability to fully understand and explain the specific internal mechanisms and reasoning of these models restricts their application in scenarios that demand high levels of transparency and interpretability.

Another constraint of this study is related to the dataset size and quality. The dataset employed for the initial exploration and training of models, while sufficient for preliminary investigations, presents potential for enhancement in terms of volume and diversity. Improving the dataset, as discussed in the Future Work section of this study, could lead to significant advancements in the model's performance and its generalizability. Expanding the dataset size and enhancing its quality are crucial for developing more robust and reliable AI-detection systems. Addressing these limitations will not only improve the efficacy of the models but also extend their practical applicability in diverse real-world contexts.

IV. Conclusion and Future Work

In this study, our primary contribution was the creation of a novel dataset of 10,000 records, comprising both human and machine-generated text. We leveraged this dataset to develop and test several models, including conventional machine learning algorithms and deep learning algorithms. We successfully deployed one of these models, capable of predicting whether a body of text is generated by a machine or a human. This contribution is crucial given the increasing prevalence of machine-generated content in online narratives, which may not always be fact-checked or scientifically accurate. We did not compare our results with existing studies, as we introduced and utilized a new dataset to apply various models. Our analysis revealed that conventional machine learning models, such as XGBoost, Logistic Regression, and Random Forest, performed robustly with traditional vectorization techniques like BoW and TF-IDF. This performance is likely due to the straightforward nature and manageable size of the dataset. In contrast, deep learning models showed a slightly lower performance. These results suggest that deep learning models require larger and more diverse datasets to fully leverage their potential for capturing contextual nuances.

In the future, we will utilize the paid GPT-4 service instead of the open-source DistilGPT-2 LLM, potentially providing more sophisticated and accurate outputs. Expanding and refining the dataset could also yield improvements. For instance, incorporating more diverse datasets could address issues related to English as a second language writing styles, thereby achieving more authentic linguistic diversity. Increasing the dataset size could further enhance model performance, providing more comprehensive training data for both conventional and deep learning models. Additionally, future research could streamline the current manual process of crafting prompts for the AI model by automating prompt creation. This automation could significantly improve the efficiency and scalability of the data generation process. Furthermore, using Wikipedia articles

published before ChatGPT's release in November 2022 would ensure that all human-labeled data are genuinely human-generated. Finally, exploring faster data collection methods, such as obtaining multiple records in bulk per API request, could expedite the process and improve the overall efficiency of data acquisition.

REFERENCES

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

[2] 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.

[3] 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.

[4] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *Advances in neural information processing systems*, vol. 32, 2019.

[5] 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.

[6] C. Chaka, "Detecting ai content in responses generated by chatgpt, youchat, and chatsonic: The case of five ai content detection tools," *Journal of Applied Learning and Teaching*, vol. 6, no. 2, 2023.

[7] W. Liang, M. Yuksekgonul, Y. Mao, E. Wu, and J. Zou, "Gpt detectors are biased against non-native english writers," *arXiv preprint arXiv:2304.02819*, 2023.

[8] 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.

[9] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson, "The curse of recursion: Training on generated data makes models forget," *arXiv preprint arxiv:2305.17493*, 2023.

[10] H. Face, "Distilgpt2." [Online]. Available: https://huggingface.co/distilgpt2

[11] ——, "Transformers." [Online]. Available: https://github.com/huggingface/transformers/tree/main/examples/research_projects/distillation

[12] 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.

[13] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 20, no. 2, pp. 215–232, 1958.

[14] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[15] 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*, 2016, pp. 785–794.

[16] A. Al Bataineh and D. Kaur, "Immunocomputing-based approach for optimizing the topologies of lstm networks," *IEEE Access*, vol. 9, pp. 78 993–79 004, 2021.

[17] A. Al Bataineh and S. Manacek, "Mlp-pso hybrid algorithm for heart disease prediction," *Journal of Personalized Medicine*, vol. 12, no. 8, p. 1208, 2022.

[18] 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.

[19] A. Al Bataineh, D. Kaur, and S. M. J. Jalali, "Multi-layer perceptron training optimization using nature inspired computing," *IEEE Access*, vol. 10, pp. 36 963–36 977, 2022.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

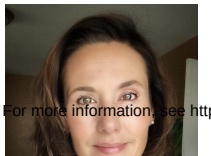
[23] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[24] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

Ali Al Bataineh is the Director of the Artificial Intelligence Center and an Assistant Professor of Electrical and Computer Engineering at Norwich University. Holding a Ph.D. in Electrical Engineering with a focus on artificial intelligence, Dr. Al Bataineh's research interests encompass AI-driven optimization, natural language processing, and machine learning applications in both academic and industrial settings. He has collaborated on multiple grants from the NSF, NIH, Department of Education, and other organizations, advancing research in AI and related fields.

Rachel Sickler is a Senior Developer and Machine Learning Engineer at Norwich University Applied Research Institutes. Rachel's experience includes software engineering, business analytics, natural language processing, predictive analytics, model observability and explainability, end-to-end data engineering and machine learning engineering. Her current focus areas are social cybersecurity , criminal justice and AI ethics.

Kerry Kurcz is a Senior Data Scientist at Norwich University Applied Research Institutes (NUARI) where she leads NUARI's Natural Language Processing (NLP) research. She has published articles in various Machine Learning subjects, including NLP, Topic Modeling, and Classification.



Kristen Pedersen is the Vice President and Chief Research officer at the Norwich University Applied Research Institutes. She is responsible for NUARI's cyber operations and portfolio of contracts and research projects for DHS, DOD, NSA, and national critical infrastructure organizations. Kristen's research areas include influence/information opera-