The use of Machine Learning Algorithms to distinguish between Large Language Model Generated Articles and Human written Articles

Kkabir Raj Bhalla

Received March 24, 2023
Accepted June 29, 2024
Electronic access July 15, 2024

With the advent of the age of artificial intelligence and widespread use of LLMs (large language models) for a variety of tasks (including content generation of images and text), the use of LLMs by malicious users for nefarious purposes is a problem that could pose a big threat in the future. LLM generated articles, specifically, could be used for malicious purposes such as spreading of misinformation or cause other issues such as the use of LLM in breach of academic integrity. Our work adds to current literature in this field as no other research has focused on the issue of LLM generated articles detection specifically. Our work focuses on the same and utilizes a new novel independently collected dataset of humans and LLM generated articles. Through the use of a large corpus of data and the use of multiple machine learning algorithms, we demonstrated the promise and ability of these algorithms to classify human generated articles and LLM generated articles. We found all of the algorithms to be very accurate in classifying the articles, with us obtaining an accuracy of 0.990 and F1-scores and ROC AUC of over 0.980. In addition, we tested the models trained on data generated by GPT 3.5 on Google's Gemini and GPT 4.0, demonstrating it is possible to generalize detection models trained on Chat-GPT articles to other LLMs. We conclude that it is possible to use machine learning algorithms to classify LLM generated articles compared to those written by humans and provide insight into the use of machine learning algorithms for classification of LLM generated text in general due to the high accuracy of the models we trained.

Keywords: Large language models, classification, machine learning, Chat-GPT, Gemini

Introduction

Throughout recent times, society has seen an influx of interest in computer generated text through the use of Large language models (LLMs), driven in part by the revolutionary capabilities and success of Open-AI's Chat-GPT allowing for a multitude of applications in areas including education [1], programming [2], and academia. The ease of access of these models has had a profound effect on the world so far allowing anyone, regardless of circumstances, the ability to interact with and use LLMs for purposes such as teaching material undergraduate courses [1] and bug and error correction in programming [2].

However, at the same time many concerns have been raised about the widespread misuse of LLMs including their use in academia, specifically the violation of academic integrity. While the positive impact of LLMs in an academic setting has been made abundantly clear. (Cotton et, al., (2024) [3] discuss an increase in engagement of students through creation of personalized assessments as well as remote learning powered by chat-APIs), the risk of plagiarism still persists. LLMs generated texts based on prompts provided by users and could be used maliciously if students submit works generated by LLMs for assignments. Cotton et, al., (2024) [3] describes the potential implications of the submission of AI generated text as causing difficulty for academic staff in assessing a student's true understanding of the topic, unfairness due to the fact that not all students may have access to the same models, and overtime a "devaluation of degrees".

Furthermore, the possibility of using LLMs to contain inaccuracies in generated text may lead to misinformation. Beutel et, al., (2023) described the use of ChatGPT in requesting summaries of different research papers and mentioned "incorrect information" in the results returned by the model as well as differences within the results themselves when ChatGPT was prompted to regenerate the response [4]. This demonstrates the potentially unreliable nature of LLMs as they may occasionally contain inaccurate information and perhaps lead to disastrous consequences if these inaccuracies are not identified in time.

In addition, LLMs could also be used to intentionally spread misinformation. While filtering systems exist on publicly available LLMs such as Open-AI's Chat-GPT and Google's Gemini in an attempt to prevent malicious users from generating inappropriate and harmful content, it is still possible to bypass these restrictions through a variety of methods. Discussed the use of adversarial attacks on LLMs and are able to successfully generate objectionable from a range of LLMs including Chat-GPT, Bard and open source LLMs such as LLaMA-2-Chat and Pythia [5]. Such attacks can have devastating consequences when

© The National High School Journal of Science 2024

one considers potential uses such as generating fake new articles to deliberately spread misinformation or push a political agenda.

Thus, in light of these issues, this paper seeks to find a way to differentiate LLM generated articles compared to ones written by human authors as a way of responding to the aforementioned issues.

In this paper we use a large dataset of independently collected human written articles and LLM generated ones and make use of multiple machine learning algorithms to try to distinguish between human and LLM written articles. We find that each algorithm performs to a high degree of accuracy and that Logistic regression, Support Vector Machines and Multi-Layer perceptron neural networks perform distinctly better than Random Forest and AdaBoost in terms of both test accuracy and F-1 Score.Next we discuss the literature review, followed by the discussion of results, the methodology used, limitations and then our conclusion.

Literature Review & Past Works

Generative Artificial Intelligence

Generative artificial intelligence (AI) refers to a branch of machine learning which involves the use of algorithms and models to generate content (often in the form of images, videos, audio, or text) that is almost indistinguishable to those created by humans.

The most commonly used generative AIs include those used for image generation (such as OpenAI's DALLE models) which make use of diffusion models and text generation. Discuss the use of diffusion models in image synthesis and conclude they are able to achieve higher quality results in image synthesis than previous "state-of-the-art" generative adversarial models (at the time the paper was written)[6].

For the purposes of this paper, we are focused and interested on the text generation subsection of generative AI specifically through the use of Large Language Models.

Large Language Models

Large language models (LLMs) are a type of deep learning algorithm based upon the idea of a transformer model introduced in Waswani et, al., (2017)[7]. Transformers are a type of neural network responsible for revolutionizing natural language processing (NLP) that make use of self-attention mechanisms (a method of weighing the importance of different elements in a sequence) allowing for it to recognize long range dependencies and contextual clues.

LLMs are built upon the aforementioned transformer architecture and are pre-trained using huge amounts of data (traditionally done through the use of unsupervised machine learning techniques).

The large language model that we are going to be using in order to train the classification models is going to be the GPT-3.5-turbo, a variation of the Chat-GPT LLM owned by OpenAI. This, along with additional details about the model, are discussed in the methodology and data section.

Past Works

The topic of differentiating between AI-Generated text and human isn't a novel one. Alon & Kamfonas (2023)[8] Attempt to filter through adversarial suffix attacks[5] by calculating the perplexity values of prompts to differentiate between normal prompts and those with adversarial attacks. They found that simply using perplexity filtering was subject to many false positives and trained a GBM model on factors like perplexity and token length to resolve this issue.

Chen et, al., (2023)[9] discuss the difficulty in detecting LLM-generated misinformation and conclude it as being more difficult for humans and detectors to detect.

Hayawi et, al., (2024)[10] discuss the use of machine learning algorithms(including random forest, logistic regression and support vector machines) along with Long Short-Term Memory (a kind of Recurrent Neural Network) to classify multiple genres of text as generated by either human, BARD (an LLM run by google), or GPT (text generated by Chat-GPT). They used a wide variety of text types including: essays, poems, stories, and python code.

For the essay data set they found Random Forest (RF) to be the most accurate (95.74%) and precise (0.87) for the essay classification and Support Vector Machines (SVM) to be the most accurate (88.73%) for the story classification. This data acts as proof of concept of the fact that it is possible to classify large textual data as either human or LLM generated through the use of machine learning algorithms.

Due the accurate nature of RF and SVM is detecting essays and stories (both longer forms of text like articles), we will include these along with other models to compare their accuracy and see if they will be as useful in detecting LLM generated articles. They also that while the BARD and Human stories were classified very effectively, all of the models struggled with GPT. This may provide future issues with our work as articles may be similar to stories due to their lengths and possibly the style they are written in.

Our work differs from what's been done in the past in these regards. Firstly, no previous works delving into a similar topic (e.g. Hayawi et, al., (2024)[10]) has tackled the issue of identifying LLM generated news articles, sticking only to code and other text. It is important to address this gap both due to the possibility of academic dishonesty/plagiarism as well as possible propaganda creation through LLM articles. Secondly, we have independently collected a new and novel dataset of a multitude of articles created by both LLMs and humans. Our dataset was

© The National High School Journal of Science 2024

completely balanced as it was trained and tested on the same number of articles created by both LLMs and humans, with a huge number of articles (over 2000 each for LLMs and humans) in contrast to other works such as Hayawi et, al., (2024) [10] which had a much more imbalanced dataset. Finally, we made use of a multitude of both classical machine learning as well as deep learning models to gauge the difference and see if there would be an observable difference between them.

Results & Discussion

Machine Learning Model Evaluation and Discussion

In this section we detail the results of the models in classifying Chat-GPT generated and human written articles.

For AdaBoost we tested the number of estimators and learning rate of the model, with our results being shown in Table 1. As expected, the model with the highest number of estimators (50) and slowest learning rate (1.0) performed the best in terms of train and test accuracy (0.957 and 0.953 respectively). This is also true for the F1-Score and ROC AUC score, with both being 0.952 with a small standard deviation of 0.007, suggesting AdaBoost is accurate in determining all LLM vs Human generated articles as well as not misidentifying each as the other. Due to the small difference in accuracy for all hyperparameters, no underfitting or overfitting was observed for any of the hyperparameter combinations. Overall, AdaBoost follows the expected pattern in all of the metrics(ie recall, precision, F1 Score, and ROC AUC) in which having a higher number of estimators leads to higher metric score and that while comparing two models with the same number of estimators, the one with the lower learning rate is more accurate and precise.

The hyperparameters we changed for RF included the number of estimators as well as the criteria on how to split data (the results are shown in Table 2). As expected, with a higher number of estimators, the higher training accuracy rate for both Gini and Log loss. This was however not true for the F1 Score and ROC AUC (and through the way they are calculated, recall and precision score). Decreasing the number of estimators from 100 to 50 for Log loss demonstrated an increase in both the F1 Score and ROC AUC, suggesting that perhaps a higher number of estimators may make the model more complex and as such possibly inaccurate in some instances. Furthermore, it is interesting to note that while both of Gini's test accuracies were higher that its train accuracies, the opposite was true for Log loss. This may suggest that Gini is a model more prone to underfitting than Log loss which is more prone to overfitting, leading to a conclusion that it may be more effective to use Gini for classification tasks for random forest.

The results for LR are displayed in Table 3. The hyperparameters we tested for LR included the algorithm used to determine the penalty, with us testing L2 and no penalty. Using L2 demonstrated the most effective model, out performing the model with no penalty in all metrics with a test accuracy of 0.986 (vs 0.983) and test accuracy of 0.990 (vs 0.978). In addition, using L2 demonstrated a test accuracy that was higher than the train accuracy, suggesting underfitting while using no penalty demonstrated a test accuracy lower than the train accuracy, suggesting overfitting during training.

Our results for SVM models are demonstrated in Table 4. For SVMs, we tested changing the kernel type between RBF, linear, poly, and sigmoid. Our results demonstrated that using a linear kernel type demonstrated the most efficient model with the highest training accuracy (0.984), testing accuracy (0.988), and ROC AUC score (0.988). There was a noticeable but small decrease in the metrics between linear and the RBF and poly kernel types, but sigmoid was shown to be significantly worse than the other 3 kernel types, with the lowest accuracy in both training and testing as well as ROC AUC and F1 score. It is also interesting to note how linear was the only kernel type with a demonstrated underfitting as its test accuracy was higher than its train accuracy. Overall, linear was demonstrated to be the best SVM kernel type for classification.

For MLP neural networks, the hyperparameters we tested included 2 weight optimisation solvers as well as 2 different neural network structures (the results are showcased in Table 5). Our results demonstrate that the "adam" optimizer has both the highest train and test accuracy, highest train accuracy with a hidden layer size (50,50) being 0.987 and highest test accuracy being 0.987 with the hidden layer size (100,100). Overall, there was a near negligible difference between the different optimizers and hidden layer sizes, except for train accuracy in which "adam" was demonstrated to be better than "lbfgs".

Table 6 demonstrates the highest test accuracy of each model. We are able to observe that each model is able to perform to a high degree of accuracy when detecting LLM generated articles versus those written by humans. LR, SVM, and MLP are demonstrated to have been particularly accurate in these detections with over 98% accuracy rate and an almost negligible difference between them.

Our results suggest that ensemble type models (such as Adaboost and RF) are less suited to classification of LLM generated articles compared to the classification algorithms such as LR, SVMs and MLP neural networks. This is further demonstrated due to the fact that all three of LR, SVM and MLP significantly outperformed AdaBoost and RF in terms of the recall score, precision score, F1-score and ROC AUC.

LR is demonstrated to be decisively, albeit by a small amount, better than SVM and MLP due to it having a higher score in all metrics (except in Recall score which it tied with SVM for 0.991). This is further supported due to the fact it has the smallest standard deviation for all test accuracy, F1-Score and ROC AUC suggesting it may be the optimal algorithm for such tasks.

Train accuracy ± std	Test accuracy ± std	Recall score ± std	Precision score ± std	F1 Score ± std	ROC AUC ± std	N-estimators	Learning rate
0.957 ± 0.004	0.953 ± 0.006	0.944 ± 0.014	0.959 ± 0.004	0.952 ± 0.007	0.952 ± 0.007	50	1
0.946 ± 0.009	0.943 ± 0.005	0.948 ± 0.010	0.938 ± 0.007	0.943 ± 0.004	0.944 ± 0.005	30	1
0.912 ± 0.006	0.922 ± 0.008	0.917 ± 0.010	0.928 ± 0.010	0.922 ± 0.008	0.922 ± 0.008	10	1
0.952 ± 0.005	0.948 ± 0.004	0.950 ± 0.006	0.944 ± 0.009	0.947 ± 0.005	0.948 ± 0.004	50	1.5
0.934 ± 0.004	0.942 ± 0.010	0.935 ± 0.011	0.945 ± 0.012	0.940 ± 0.010	0.941 ± 0.010	30	1.5
0.904 ± 0.004	0.903 ± 0.001	0.901 ± 0.016	0.910 ± 0.009	0.906 ± 0.004	0.904 ± 0.001	10	1.5

Table 1 AdaBoost results

Train accuracy ± std	Test accuracy ± std	Recall score ± std	Precision score ± std	F1 Score ± std	ROC AUC ± std	N-estimators N-estimators	Criteria Criteria
0.955 ± 0.006	0.959 ± 0.003	0.963 ± 0.005	0.954 ± 0.007	0.959 ± 0.002	0.959 ± 0.003	100	Gini
0.954 ± 0.005	0.957 ± 0.008	0.957 ± 0.009	0.956 ± 0.009	0.957 ± 0.008	0.957 ± 0.008	50	Gini
0.959 ± 0.001	0.954 ± 0.001	0.957 ± 0.001	0.950 ± 0.002	0.953 ± 0.001	0.954 ± 0.001	100	Log loss
0.956 ± 0.002	0.953 ± 0.006	0.961 ± 0.008	0.955 ± 0.007	0.958 ± 0.006	0.959 ± 0.006	50	Log loss

Table 2 Random forest results

Train accuracy ± std	Test accuracy ± std	Recall score ± std	Precision score ± std	F1 Score ± std	ROC AUC ± std	Penalty
0.986 ± 0.004	0.990 ± 0.002	0.991 ± 0.004	0.989 ± 0.004	0.990 ± 0.002	0.990 ± 0.002	L2
0.983 ± 0.003	0.978 ± 0.002	0.977 ± 0.004	0.979 ± 0.005	0.978 ± 0.003	0.978 ± 0.002	None

Table 3 Logistic regression results

Train accuracy ± std	Test accuracy ± std	Recall score ± std	Precision score ± std	F1 Score ± std	ROC AUC ± std	Kernel type
0.983 ± 0.004	0.983 ± 0.005	0.980 ± 0.005	0.985 ± 0.004	0.982 ± 0.004	0.983 ± 0.005	RBF
0.984 ± 0.003	0.988 ± 0.003	0.991 ± 0.003	0.986 ± 0.003	0.988 ± 0.003	0.988 ± 0.003	linear
0.979 ± 0.002	0.978 ± 0.004	0.987 ± 0.005	0.970 ± 0.008	0.979 ± 0.004	0.978 ± 0.004	poly
0.915 ± 0.007	0.900 ± 0.005	0.912 ± 0.005	0.891 ± 0.005	0.901 ± 0.005	0.900 ± 0.005	sigmoid

Table 4 Support vector machine results

Train accuracy ± std	Test accuracy ± std	Recall score ± std	Precision score ± std	F1 Score ± std	ROC AUC ± std	size size	Optimizer Optimizer
0.983 ± 0.002	0.986 ± 0.004	0.984 ± 0.007	0.987 ± 0.006	0.985 ± 0.004	0.986 ± 0.004	(50,50)	lbfgs
0.983 ± 0.002	0.986 ± 0.005	0.988 ± 0.008	0.984 ± 0.004	0.986 ± 0.005	0.986 ± 0.005	(100,100)	lbfgs
0.984 ± 0.003	0.987 ± 0.003	0.988 ± 0.005	0.986 ± 0.005	0.987 ± 0.003	0.987 ± 0.003	(50,50)	adam
0.987 ± 0.001	0.986 ± 0.002	0.986 ± 0.001	0.985 ± 0.004	0.986 ± 0.002	0.986 ± 0.002	(100,100)	adam

Table 5 Multi-layer perceptron neural network results

Evaluation of Accuracy of GPT-Trained Models Being Applied to Other LLMS And Assessment of Generalizability

Table 7 summarizes the results of training our model on ChatGPT 3.5 and using it to test articles generated by humans and Google's Gemini LLM. All of the models performed extremely well in the recall score, with the lowest and highest being 0.976 and 0.998 respectively. This demonstrates how it is very unlikely that human generated articles will get mis-identified as being

© The National High School Journal of Science 2024

Test accuracy ± std	Recall score ± std	Precision score ± std	F1-Score ± std	ROC AUC ± std	Model
0.953 ± 0.006	0.944 ± 0.014	0.959 ± 0.004	0.952 ± 0.007	0.952 ± 0.007	AdaBoost
0.959 ± 0.003	0.963 ± 0.005	0.954 ± 0.007	0.959 ± 0.002	0.959 ± 0.003	RF
0.990 ± 0.002	0.991 ± 0.004	0.989 ± 0.004	0.990 ± 0.002	0.990 ± 0.002	LR
0.988 ± 0.003	0.991 ± 0.003	0.986 ± 0.003	0.988 ± 0.003	0.988 ± 0.003	SVM
0.987 ± 0.003	0.988 ± 0.005	0.986 ± 0.005	0.987 ± 0.003	0.987 ± 0.003	MLP

Table 6 Highest test accuracy results for all models

Test accuracy ± std	Recall score ± std	Precision score ± std	F1-Score ± std	ROC AUC ± std	Model
0.760 ± 0.017	0.976 ± 0.005	0.682 ± 0.017	0.803 ± 0.011	0.759 ± 0.017	AdaBoost
0.758 ± 0.005	0.991 ± 0.008	0.677 ± 0.004	0.804 ± 0.004	0.757 ± 0.005	RF
0.758 ± 0.004	0.996 ± 0.002	0.675 ± 0.003	0.805 ± 0.003	0.757 ± 0.004	LR
0.756 ± 0.012	0.996 ± 0.002	0.674 ± 0.011	0.804 ± 0.007	0.755 ± 0.012	SVM
0.756 ± 0.007	0.998 ± 0.001	0.673 ± 0.007	0.804 ± 0.005	0.755 ± 0.007	MLP

Table 7 Results of using GPT 3.5 trained models to classify Gemini generated articles

Test accuracy ± std	Recall score ± std	Precision score ± std	F1-Score ± std	ROC AUC ± std	Model
0.916 ± 0.013	0.971 ± 0.008	0.875 ± 0.015	0.921 ± 0.011	0.916 ± 0.013	AdaBoost
0.904 ± 0.002	0.993 ± 0.002	0.844 ± 0.002	0.912 ± 0.002	0.904 ± 0.002	RF
0.972 ± 0.001	0.995 ± 0.004	0.952 ± 0.005	0.973 ± 0.001	0.972 ± 0.001	LR
0.971 ± 0.004	0.997 ± 0.002	0.949 ± 0.006	0.972 ± 0.004	0.971 ± 0.004	SVM
0.972 ± 0.005	0.998 ± 0.001	0.949 ± 0.01	0.973 ± 0.005	0.972 ± 0.005	MLP

Table 8 Results of using GPT 3.5 trained models to classify GPT 4.0 generated articles

written by LLMs. The precision scores were slightly lower than those described in the previous section (the highest being 0.0.682 for AdaBoost), but they are still fairly adequate and definitely demonstrate both promise and proof that it is indeed possible to generalize models trained on Chat-GPT to other LLMs. This is further demonstrated due to the fact that there was a relatively high accuracy around 0.75 and 0.76 as well as a high F1-Score (the highest being 0.804 and lowest being 0.803).

Table 8 summarizes the results of training our model on Chat-GPT 3.5 and using it to test articles generated by GPT 4.0 and those written by humans. All of the models performed extremely well boasting both high accuracies (0.972 being the greatest) and high F1-Score and ROC AUC (the highest being 0.973 and 0.972 respectively). Due to the fact that each version of Chat-GPT is trained independently using more up to date data each time, as well as having a different architecture, we have demonstrated that it is indeed possible to generalize models trained on GPT generated articles to other LLMs, which have both differing architecture and data (such as GPT 4.0) and different methods of training (such as Gemini).

Furthermore, due to both the high recall rate (demonstrating it is able to detect all human generated articles and thus very unlikely to classify an article written by a human as LLM generated) and the relatively high accuracy and ROC AUC scores in Table 7 and Table 8, we are able to conclude that it is possible to generalize GPT trained detection models on other LLMs. Those with different data, architecture and methods of training (like Google's Gemini and GPT 4.0).

However, even though the results may suggest that GPT trained models would be effective classifiers for LLMs of other architecture types and training data, it is important to consider possible limitations when assessing this generalizability. Some particular LLMs may more accurately mimic writing styles of a human causing potential false positives to occur (classifying LLM generated articles as those written by humans).

This could also be true when the LLMs are trained on data significantly different to that trained by Chat-GPT. Still, it is unlikely that human written articles would be unjustly identified as LLM generated due to the fact that our dataset should provide a representative sample of human writing patterns and as such the models should be able to identify human written articles as written by humans.

Methodology

Article collection and filtering

The final dataset processed consists of 4854 articles (2427 written by human authors and 2427). The human written articles were scrapped from a variety of news sources using a Google News web scraper.

To ensure validity of the classifier, only articles written before 2019 were considered (to ensure none of the articles were written in any part by an LLM) and short articles (defined as those with fewer than 500 characters) were also discarded.

The LLM generated articles were generated by Chat-GPT 3.5 Turbo using the Open-AI API, with the LLM being prompted to write an article with the title of each filtered article (collected by the method detailed above). The same was conducted for Chat-GPT 4.0 using the OpenAI API and Google's Gemini LLM using the Gemini API and Google Cloud shell.

All of the models were trained using the GPT 3.5 generated articles, but we also tested the accuracy of these models when tested against articles generated by Google's Gemini LLM and GPT 4.0. We used these two models due to the fact that they are the largest and most widely commercially used.

Article Embedding

Before text can be processed through machine learning algorithms, it must be present in a numerical form so that the algorithms can be applied onto it. This process is referred to as embedding. We are going to be exploring the use of Sentence-BERT for this purpose. Sentence-BERT (SBERT) (introduced in Reimers, and Gurevych (2019)[11]) is a modified version of the pretrained BERT network to produce meaningful sentence embeddings. We used SBERT through the spaCy library, specifically the "en_stsb_roberta_large" model that produces vectors of dimension 1024.

Our reasoning behind choosing SBERT was due to its increased performance and quality of its embeddings, demonstrated by its high evaluation by SentEval compared to other models such as Universal Sentence Encoder and GloVe models11. In addition, it is much more computationally efficient compared to embedders like InferSent and Universal Sentence encoder. 11 demonstrated how SBERT was able to embed 2042 sentences per second on GPU versus the latter which were only able to embed 1318 and 1875 respectively.

For each article, we calculated the embedding for each sentence (stored as a 1-dimensional numpy array with 1024 elements) before calculating the mean of all the different sentence embedding for each article. This was then added to another numpy array before being stored in a CSV file.

Machine Learning algorithms used

In our evaluation we used 5 algorithms: Random forest (RF), AdaBoost, Logistic regression (LR), Support vector machines (SVM), and a multi-layer perceptron neural network (MLP) that uses backpropagation.

The reasoning behind us choosing the classical machine learning algorithms(RF, AdaBoost, LR, and SVM) was due to the fact that they have been in past works with success in similar situations and thus could be suited to the task at hand (e.g. 10). The deep learning neural network MLP was used to gauge the difference if any of using deep learning models for binary classification compared to the traditional classical methods.

Random forest: RF is an ensemble machine learning algorithm that involves the creation of multiple decision trees while it is being trained, with the most frequent output predicted by the trees being the output of the model for classification tasks (like the one considered in this paper). This reduces the impact of overfitting or bias in the data as the prediction is based on the output of multiple decision trees.

AdaBoost: AdaBoost is an ensemble type machine learning algorithm which uses the principle of boosting to combine several weak classifiers (e.g decision trees) into a single strong classifier. It sequentially trains each decision tree and assigns incorrect predictions from a previous tree a higher weight when input into the next tree.

Logistic regression: LR is a type of classification algorithm that uses the features of the input data and probability of it being a certain class using the sigmoid function (that maps any real value to a value between 0 and 1). Since the output is a value between 0 and 1, it makes logistic regression useful for binary classification tasks like the one we are considering. The parameters of the sigmoid function are estimated to best fit the data through the use of a loss function (the parameters are best fit when the loss function is minimized).

Support vector machines: SVM is a classification machine learning algorithm involving the use of each input object being plotted on an n-dimensional plane. Hyperplanes are consequently drawn to separate the two classes such that all points of both classes are on one side of the hyperplane. SVM tries to optimize the hyperplane to find one where the distance between the points and the hyperplane is maximized.

Multi-layer perceptron neural network: MLP is a type of neural network that involves multiple hidden layers each having individual perceptron neuron cells. It functions through each cell taking a certain amount of inputs, multiplying them by a weight and then adding a bias. This is then passed through a hyperbolic tan function. The MLP classifier used in our work makes use of backpropagation to train the model. This involves after the initial forward pass and error calculator, each neuron is assessed for the magnitude it contributed to the error and then consequently the weights are updated.

© The National High School Journal of Science 2024

Metrics used for evaluation

This section details some metrics we are using for model evaluation and comparison.

Train accuracy: A measure of the accuracy of the model in training through cross validation

Test accuracy: A measure of how accurate the model was when calculated as the ratio of articles classified correctly by the model divided by the total number of articles.

Recall score: A measure of how accurate the model is in determining all of the target class, calculated by tp/(tp + fn) where tp is the number of true positives and fn is the number of false negatives. This measure determines how accurate the model is in determining all of the human generated articles.

Precision score: A measure of how accurate the model is in prediction of the target class, calculated as tp/(tp + fp) where tp is the number of true positives and fp is the number of false negatives. This measure determines how accurate the model was when determining an article as human generated.

F-1 score: It is the harmonic mean of the precision score and the recall score, calculated as (precision * recall)/(precision + recall) where * is the multiplication operator

ROC area under curve: This is the area under the ROC curve and is used to guage the trade off between true positives and false positives. A higher ROC AUC value(closer to 1) demonstrates a model's ability to correctly identify true positives.

Processing parameters

In order to process the data and parameters, we make use of the Scikit-learn package to import pre-implemented versions of RF, AdaBoost, LR, SVMs and the MLP neural network. The reasoning behind us choosing our classical machine learning algorithms was that they were already used successfully in the past for similar applications (like they were used to classify long

The data was handled through the use of the Pandas library's read_csv function to load it and converted into a Numpy array for easier manipulation and handling. The data was split in a 80-20 fashion with 80% being used to train the models and 20% being used to test it. We also labeled the human written articles with the label "1" and LLM written articles with the label "0".

During the training of our models, we calculated the training accuracy using a cross validation score which trained the model in a round robin fashion, with the training data being split into 5 parts.

The hyperparameters we tested were as follow:

AdaBoost: We tested the number of estimators (number of learners after which boosting is terminated) from 50 (default) to 30 and 10, and tried two learning rates 1.0 and 1.5.

RF: We tested 100 (default) and 50 for number of estimators (trees) along with 2 different criteria (to measure the quality of a split): Gini (default) and log-loss.

LR: We tried 2 penalty types for LR including l2 (default ridge regression which adds square magnitude as penalty in the loss functions) and None (no penalty).

SVM: We tried 4 kernel types for the SVM model including RBF (default Gaussian kernel), linear (uses dot product of the input samples), polynomial (using a different notion of similarity using a polynomial function of d degrees), and sigmoid (using the hyperbolic function tanh).

MLP: We changed the hidden layer sizes and the solver used for weight optimization of the function.The layer sizes we used included (50,50) and (100,100) and we made use of the "lbfgs"(default) and "adam" optimizers. All of the models were trained with a constant 200 epochs.

Limitations

This work is not without its limitations. In training our machine learning model, we used 4854 articles with 2427 being written by human authors on a variety of subjects between 2011 and 2018 inclusive. While the range and amount of data is quite substantial to the point that we were able to train the algorithm to accurately classify our test and train data, a larger sample size made up of articles written over a larger period of time could lead to a model that is trained to be more generalizable to a range of different writing styles based on time.

Furthermore, all of the articles that were used as part of our dataset were English articles. With the ability of certain LLMs to speak in a range of languages it is possible that malicious users may use LLMs to generate such articles in other languages, in which case the fact that we only tested the models on English articles will be a limitation.

Finally, we only considered one method of embedding data using SBERT as the embedder.

Conclusion & Future Work

To conclude, this paper effectively makes a step forward concerning the field of classification of LLM generated content (specifically articles) from human ones. This work makes use of a significant dataset of human written articles as well as those generated by LLMs. We demonstrate the aptitude of multiple machine learning algorithms to distinguish LLM generated and human written articles. Our results demonstrate that although LLMs are making exemplary progress, machine learning algorithms are able to classify long text articles generated by LLMs. In addition, we showcase the potential use of these models trained on Chat-GPT generated data being applied to other LLMs, demonstrating that it is possible to generalize models trained on Chat-GPT for use to classify other LLMs.

Our work creates an avenue of research for future work including the work that may make use of other algorithms for the

same purpose of classification of human and LLM generated articles and could set precedent of general classification of long text content (such as stories, essays, and articles) as human or LLM generated. These works could directly assist in practical social impacts such are creation of stronger more robust LLM generated text detectors which can be used in multiple situations like to detect plagiarism in an academic setting or decrease the effect of misinformation in user written articles and blogs (written possibly for malicious purposes like spam or propaganda) submitted to websites.

References

1 I. Kostka and R. Toncelli, *TESL-EJ*, 2023, **27**, year.

2 N. Surameery and M. Shakor, *International Journal of Information Technology Computer Engineering (IJITC)*, 2023, **3.01**, 17–22.

3 D. Cotton, P. Cotton and J. Shipway, *Innovations in Education and Teaching International*, 2024, **61**, 228–239.

4 G. Beutel, E. Geerits and J. Kielstein, *Critical Care*, 2023, **27**, 148.

5 A. Zou, Z. Wang, J. Kolter and M. Fredrikson, *arXiv preprint arXiv:2307.15043*, 2023.

6 P. Dhariwal and A. Nichol, *Advances in neural information processing systems*, 2021, **34**, 8780–8794.

7 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, NIPS, 2017.

8 G. Alon and M. Kamfonas, *arXiv preprint arXiv:2308.14132*, 2023.

9 C. Chen and K. Shu, *arXiv preprint arXiv:2309.13788*, 2023.

10 K. Hayawi, S. Shahriar and S. Mathew, *Journal of Information Science*, 2024, 01655515241227531.

11 N. Reimers and I. Gurevych, *arXiv preprint arXiv:1908.10084*, 2019.

© The National High School Journal of Science 2024