J-Guard: Journalism Guided Adversarially Robust Detection of AI-generated News

Tharindu Kumarage Amrita Bhattacharjee Djordje Padejski Kristy Roschke
Dan Gillmor Scott Ruston Huan Liu Joshua Garland
Arizona State University
{kskumara,abhatt43,padejski,carver,dan,scott.ruston,huanliu,jtgarlan}@asu.edu

Abstract

The rapid proliferation of AI-generated text online is profoundly reshaping the information landscape. Among various types of AI-generated text, AI-generated news presents a significant threat as it can be a prominent source of misinformation online. While several recent efforts have focused on detecting AI-generated text in general, these methods require enhanced reliability, given concerns about their vulnerability to simple adversarial attacks. Furthermore, due to the eccentricities of news writing, applying these detection methods for AI-generated news can produce false positives, potentially damaging the reputation of news organizations. To address these challenges, we leverage the expertise of an interdisciplinary team to develop a framework, **J-Guard**, capable of steering existing supervised AI text detectors for detecting AI-generated news while boosting adversarial robustness. By incorporating stylistic cues inspired by the unique journalistic attributes, **J-Guard** effectively distinguishes between real-world journalism and AI-generated news articles. Our experiments on news articles generated by a vast array of AI models, including ChatGPT (GPT3.5), demonstrate the effectiveness of **J-Guard** in enhancing detection capabilities while maintaining an average performance decrease of as low as 7% when faced with adversarial attacks.

1 Introduction

Recent advances in transformer-based generative models have led to substantial enhancements in the Natural Language Generation (NLG) capabilities of advanced conversational Artificial Intelligence (AI) systems, such as ChatGPT and BARD. These AI tools generate human-like text on a large scale by leveraging state-of-the-art (SOTA) pre-trained language models (PLMs) such as GPT 4 (OpenAI, 2023), GPT 3.5 (Ouyang et al., 2022), GPT 3 (Radford et al., 2019), OPT (Zhang et al., 2022) and Lambda (Thoppilan et al., 2022). Considering the current trend of deploying these models in services offered to the general public, we can anticipate further improvements in NLG from future models.

However, deploying such NLG-capable models for public use poses the risk of potential misuse. Adversaries can employ these models to establish harmful agendas and conduct influence operations that deceptively steer the opinions of large groups of a target populace (Shu et al., 2020; Goldstein et al., 2023). AI-generated news articles are particularly concerning, as they can cause significant damage to the information ecosystem. Malicious actors can easily prompt AI models to generate text that purports to be authentic news but contains falsified information (Shu et al., 2018; Zellers et al., 2019). To make matters worse, current models are capable of generating misinformation and factually incorrect text in large volumes at a minimal cost through APIs. A recent report[1] by NewsGuard, an organization that combats misinformation online, identified an emerging set of 49 newsbots, i.e., news and information sites, that appear to incorporate AI for news generation. Therefore, it is crucial to have computational methods to discern between AI-generated news and actual human-written news to combat the persistent challenges to the information ecosystem.

In recent years, much interesting work has been done on detecting AI-generated text (Zellers et al., 2019; Mitchell et al., 2023; Kirchenbauer et al., 2023). However, most of these methods, which we discuss under Related Works (section 5), do not explicitly focus on AI-generated news. Therefore, using these general-purpose AI text detectors to detect AI-generated news has a few challenges: 1) the unique attributes of professional journalism make news articles distinct from typical human-written text. Thus applying general AI text detection methods for AI-generated news detection could lead to

[1] https://www.newsguardtech.com/special-reports/newsbots-ai-generated-news-websites-proliferating/

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 484–497
November 1–4, 2023.

false positives that potentially damage the reputation of journalists and news organizations, and 2) existing AI text detectors are highly vulnerable to adversarial attacks, e.g., paraphrasing (Sadasivan et al., 2023; Krishna et al., 2023).

To address the above challenges, we leverage the expertise of an interdisciplinary team, which includes journalists, computer scientists, and communication scholars, to develop a framework for **J**ournalism **Gu**ided **A**dversarially **R**obust **D**etection of AI-generated News (**J-Guard**)[2]. To this end, we first studied the unique professional journalism attributes of human-written news articles' writing and publishing process. Throughout the journalism process, many stylometric cues are incorporated, including journalism standards employed by the journalist as well as specific newsroom style guides and standards imposed by the newsroom editors. Here we hypothesize that even though the PLMs learn human-level writing via pretraining, they potentially will display semantic gaps in replicating these style guides and journalism standards inherent to the news production process. Therefore, we propose incorporating a simple yet effective set of auxiliary stylistic cues to guide the existing supervised AI text detectors to discern real-world journalism with the AI generation of news articles using PLMs. Furthermore, as we will show, since these cues quantify the high-level stylometry of the text, the detection process is more robust to the character and word level perturbations, thus, reinforcing the adversarial robustness of our AI-generated news detection methodology.

To summarize, the main contributions of our paper are as follows:

1. To the best of our knowledge, we are the first to study and quantify stylistic cues resulting from the latent journalism process in real-world news organizations towards discriminating AI-generated news.

2. We propose a computational framework incorporating these stylistic cues to detect AI-generated news.

3. We conduct extensive experiments on a publicly available vast array of PLMs, including ChatGPT (GPT 3.5), to show our approach's effectiveness in detecting AI-generated news.

4. By producing character and word level attacks, we empirically show how the stylistic cues we incorporated improved the adversarial robustness of AI-generated news detection.

2 Journalism Background

Journalism as an industry does not universally subscribe to codes of conduct, owing largely to a historical rebuke of standardization as a profession (Shapiro, 2010). Several trade groups, including the Society for Professional Journalists, have created detailed style guides. Many news organizations have adopted them internally, and others have created their versions. Scholars (Broersma, 1880; Shapiro, 2010; Mateus, 2018) have noted that, though the reporting process is typically situational, which makes it difficult to routinize, there are some key areas in which common methods, processes, and values signal an intent to establish credibility. And the form and style are integral to convincing people of the 'truthiness' of newsworthy events (Broersma, 1880; Mateus, 2018).

Journalistic practices that have been widely adopted include the use of the inverted pyramid as a storytelling format (Mateus, 2018) and a style of writing based on the Associated Press (AP) Stylebook [3]. Mateus (Mateus, 2018) describes form and style "as key components of journalistic discourse that, in a given time, are able to generate credibility and confidence." Though the AP Stylebook is not universally followed among news organizations, and some make situational exceptions, if we encounter purported news articles that are widely divergent from what AP recommends, we hypothesize that this is a strong signal of inauthenticity. In fact, adherence to the Stylebook is one of the key factors in the Associated Press' automated journalism efforts (Linden, 2017).

In our study, we aim to integrate the aforementioned hypothesis of inauthenticity into the task of detecting AI-generated news. Specifically, we investigate the extent to which current AI models are capable of generating news articles that adhere to professional journalism standards. Figure 1 illustrates a clear distinction in the distribution between GPT3-generated news articles and those written by humans from reputable news organizations such as CNN and the Washington Post. As illustrative examples of journalism features, we consider the length of introductory sections (leading sentences

[2]Feature extraction and J-Guard code is available at https://github.com/TSKumarage/J-Guard.git

[3]https://www.apstylebook.com/

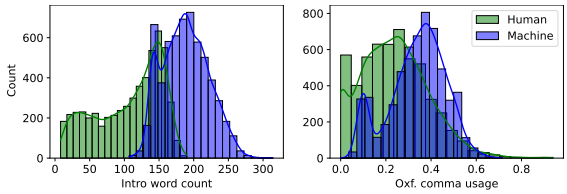

Figure 1: Distribution of GPT3 Generated News vs Human-written News.

and paragraphs) and the usage of Oxford commas. Professional journalism typically employs shorter and more concise introductions, while the use of Oxford commas is infrequent in accordance with AP standards. Hence, we observe the potential for leveraging our hypothesis to enhance the detection of AI-generated news. In the subsequent section, we will delve into a detailed discussion of the journalism features that can be utilized for detecting AI-generated news.

3 AI-generated News Detection

This section presents the details of the **J-Guard** framework. The **J-Guard** framework consists of two main components: (a) the base AI text detector component and (b) the Journalism guidance component. The base AI text detector is any PLM sequence classification model. The journalism guidance component injects auxiliary journalism cues into the detection pipeline, thus transforming the base detector into an AI-generated news detector. We will provide a comprehensive discussion of these two components in the following sections.

3.1 Base AI Text Detector

The Base AI text detector component consists of a pretrained transformer encoder stack with n encoders to learn the semantic representation of the given input news article X. Here we define $(x_1, x_2, ..., x_k)$ as the token representation of the input X according to the tokenizer of the PLM model we choose for the base detector component. We denote the representation learned by the base AI text detector as $B_{k\times d}$ where k is the sequence length (i.e., number of tokens of the input news article), and d is the hidden state size of an encoder block. From the representation matrix, $B_{k\times d}$, we select the final hidden vector representation of the special token [CLS], B_d^{CLS} as the feature vector for our task of detecting AI-generated news. Then B_d^{CLS} is passed to the journalism guidance component for further processing.

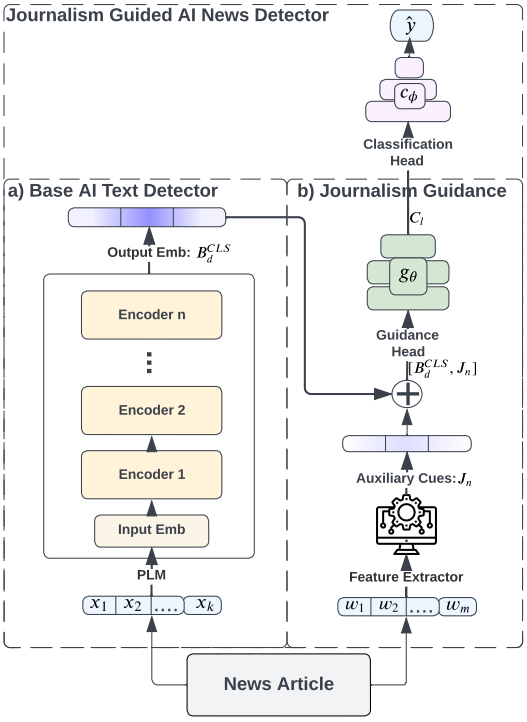

Figure 2: Proposed framework **J-Guard**: The base detector component here is a supervised PLM-based detector for AI text detection.

3.2 Journalism Guidance

The cornerstone of the **J-Guard** framework lies in the journalism guidance of the Base AI text detector toward detecting AI-generated news—different modules within the journalism guidance component help achieve this goal. As illustrated in Figure 2, the journalism guidance component comprises a Journalism Feature Extractor and a Guidance Head as sub-components.

3.2.1 Journalism Feature Extractor

As postulated in Section 2, encountering a news article that widely deviates from the recommended styles and standards of the AP stylebook may serve as a strong indication of inauthenticity. Therefore, the journalism feature extractor is a computational module that incorporates this hypothesis to enhance the detection of AI-generated news. The feature extractor takes the input news article X in the form of a set of tokens $w_1, w_2, ..., w_m$. Here, $(w_1, w_2, ..., w_m)$ represents the tokenized version of the input article X using an improved Treebank Word Tokenizer[4]. Subsequently, a set of extractor functions $f \in F$ is applied to these tokens to extract various scores that quantify the divergence of

[4]https://www.nltk.org/api/nltk.tokenize.word_tokenize.html

the input article from the AP recommended styles and standards. For discussion, it is useful to label three subsets of the extractor function set F $F^i|i \in 1,2,3$, such that $F = F^1 \cup F^2 \cup F^3$. The three subsets can be broadly defined as follows:

1. F^1: **Organization and grammar standards** - functions that quantify the wording and grammatical structure of the news article (sentences and paragraphs forming)

2. F^2: **Punctuation usage** - functions that quantify the punctuation usage of the news article

3. F^3: **Formatting standard violations** - functions that quantify the violations of the formatting of different elements in a news article, such as date, time, and number, in reference to the AP standards

Within each extractor category, we extract multiple features that can quantify the deviation of the input article from the AP recommended styles and standards. In F^1, we examine the overall wording structure of the news article, as well as the leading sentence and paragraph, as the size of these components could serve as indicators of inauthenticity. For example, a large leading or introductory part is not very common for news articles. Additionally, we consider grammatical elements, such as tense and voice, which can provide cues about journalism standards. For instance, the use of past tense and passive voice is not common in news writing. As a result, the following features are extracted: mean word count (WC), mean sentence count (SC), WC of the leading sentence, WC of the leading paragraph, mean SC with passive voice, and mean SC with past tense. In F^2, we analyze the usage of punctuation. In addition to standard punctuation marks, we also examine symbols that are rarely found in genuine news articles, such as the number sign. Consequently, the following features are extracted in F^2: mean usage of "!" "#", """, and the Oxford comma per paragraph. Lastly, under F^3, we investigate format violations in the input article based on AP standards. Specifically, we identify and count violations related to date, time, and number formats. The detailed implementations of each feature extractor function can be found in the appendix section A.

Some of the feature extractors mentioned above return mean values in the range of [0,1] while some return absolute counts, which will be larger than 1. Therefore we normalize the feature vector before incorporating it with the task of AI-generated news detection. Let n be the number of journalism features, $f_i \in F$ and $W = (w_1, w_2, \ldots, w_m)$ be the treebank tokenization of X then we define the final normalized journalism feature vector J_n as:

$$J_n = \frac{[f_1(W), f_2(W), ..., f_n(W)]}{|||[f_1(W), f_2(W), ..., f_n(W)]|||}. \quad (1)$$

3.2.2 Guidance Head

We propose to enhance the detection capabilities and adversarial robustness of our detector by incorporating the learned journalism features, J_n, into the output of the base AI text detector, B_d^{CLS}. A naive approach would be to simply concatenate both features and pass them through the fully connected feedforward neural network, which we refer to as the Classification Head, to predict the final classification label $\hat{y}$. However, this naive approach may lead to the overshadowing of J_n by B_d^{CLS} due to the large dimensionality of B_d^{CLS} compared to J_n. Furthermore, direct concatenation of the two feature vectors without considering their different ranges poses a feature scaling issue. To address these challenges, we propose the incorporation of an additional set of feedforward layers, referred to as the Guidance Head. This Guidance Head includes a hidden layer with a size equal to or larger than the input layer. This choice is made to prevent overshadowing of J_n. The Guidance Head learns the relationships between the feature vectors B_d^{CLS} and J_n, without overshadowing J_n, by mapping the input $[B_d^{CLS}, J_n]$ to a higher-dimensional feature space. Note that we first normalize the $[B_d^{CLS}, J_n]$ vector before passing it to the Guidance Head to avoid feature scaling issues. Finally, the Guidance Head's output layer produces a reduced vector of the scaled-up hidden representation, which we pass to the Classification Head for the final prediction. To summarise, as shown in equation 2, the whole purpose of Guidance Head is to learn the function g_θ that learns fusion between B_d^{CLS} and J_n.

$$C_l = g_\theta \left(\frac{[B_d^{CLS}, J_n]}{|||[B_d^{CLS}, J_n]|||} \right) \quad (2)$$

Here, C_l is the reduced vector of size l produced by the output layer of the Guidance Head.

Finally, the output of the Guidance Head, C_l is passed to the Classification Head to predict the final classification label $\hat{y}$. Using the ground truth

label, we incorporate standard cross-entropy loss to train the whole framework **J-Guard** end to end.

4 Experiments and Results

This section describes the experimental settings used to validate our framework, including the datasets and baselines, followed by a thorough analysis of the experiments. We conducted several experiments to investigate whether the proposed journalism features can improve the detection of AI-generated news. We aim to answer the following two research questions through our experiments:

- **RQ1** - Do the identified journalism features, enhance the detection of AI-generated news?
- **RQ2** - Do the identified journalism features, enhance the adversarial robustness of AI-generated news detection?

4.1 Datasets and AI Generators

We evaluate our approach on a vast array of AI generators, i.e., PLMs — To this end, we use the benchmark datasets TuringBench (Uchendu et al., 2021) and NeuralNews(N.News) (Tan et al., 2020). TuringBench is a dataset consisting of human-written news articles, mostly from CNN and the Washington Post, and AI-generated news from more than 10 PLM generators. Of these, we used the following PLMs for our analysis: Grover, CTRL, PPLM$_{gpt2}$ (base model used is GPT2), GPT2, GPT3. Within TuringBench, data is generated using various combinations of PLMs and model sizes. To maintain brevity in our analysis, we have included only the largest model size for each PLM. This selection is justified by the understanding that the largest model size for each PLM is expected to produce the highest quality text, making it more challenging to detect. Therefore, our results can be extrapolated to smaller PLMs as well. The Neural News dataset contains only news articles generated by Grover. The human-written articles included in this dataset are collected from the GoodNews dataset, which features news from the New York Times.

Furthermore, we performed our experiments on a ChatGPT dataset that we created. Given the human-like quality of text generated by newer PLMs like GPT3.5 and GPT4 (OpenAI, 2023), it is important to evaluate our detection framework on such language models. To create this dataset, we followed steps similar to the ones in the TuringBench paper (Uchendu et al., 2021). Specifically, we sampled around 9,000 news articles from CNN and the Washington Post and use these as 'human' written articles. For each of these articles, we prompt ChatGPT (with backend gpt-3.5-turbo, model version as of March 14, 2023) to generate an equivalent news article. To do this, we experiment with several types of prompts, and for the final data generation, we use the prompt: "Generate a news article with the <headline>.", where <headline> is the headline from the corresponding human written article. For the ChatGPT generations, we set top_p to 1, $temperature$ to 0.5 and limit the length of the generated text to 1024 tokens. The final dataset contains 9k human-written and 9k ChatGPT-generated articles, which we divided into train, test, and validation splits (7:2:1) similar to TuringBench. We will release this dataset to the public upon acceptance of the paper (section 8.2).

4.2 Baselines

Our experiments consist of two categories of AI news detector baselines: First, we study simple feature-based classification schemes which use logistic regression (LR) with BOW and Word2vec features as a baseline to evaluate the quality of the journalism features (JF) we selected via our journalism analysis. Second, we aim to empirically compare and validate **J-Guard** with SOTA PLM-based methods for AI-generated text detection. The SOTA baselines can be further categorized into 1) **Zero-shot PLM-based classifiers**: GLTR (Gehrmann et al., 2019), and the newer zero-shot baseline DetectGPT (Mitchell et al., 2023). These two approaches work without supervised training datasets for detecting AI-generated text and 2) **Supervised PLM-based classifiers**: We consider OpenAI's GPT-2 detector (RoBERTa-large) as our supervised PLM-based detector baseline. We considered two variants of this model i) OpenAI$_{Zero}$ - OpenAI's off-the-shelf GPT-2 detector without any task-specific tuning, ii) OpenAI$_{FT}$ - OpenAI's GPT2 detector finetuned for AI news detection. Further technical details about the baselines can be found in the appendix section B.

4.3 Detection Setup

Implementation Details of J-Guard: The base AI text detector is one of the key components of the **J-Guard** framework, involving a supervised PLM specifically designed for detecting AI-generated text. In our research, we conducted experiments using various existing PLMs (base size), including RoBERTa, BERT, DeBERTa, and DistilBERT.

Among these models, RoBERTa exhibited the highest performance, and therefore, it was selected as the base AI text detector for the **J-Guard** framework, while reporting the experiment results. Both the Classification Head and the Guidance Head were implemented using feedforward neural networks comprising one hidden layer. For the training of the overall framework, a max length of 512, a learning rate of 2×10^{-5}, and a dropout rate of 0.2 were employed. The training process utilized a 40 GB NVIDIA A100 GPU ($\approx$ 1hr per AI generator).

Task Details: We consider the task of AI-generated news detection as a binary classification problem. In our data, we have train, test, and validation (7:2:1 ratio) splits for each AI generator, where we use the train set to finetune models on the task of AI news detection and the test set to record the classification performance. The validation set was used for early stopping to determine the number of training epochs. See appendix section C for more details.

4.4 Adversarial Attack Setup

In order to validate the adversarial robustness of the detector, we conducted two common attacks that have been observed in previous work: Cyrillic injection and paraphrasing (Crothers et al., 2022; Sadasivan et al., 2023; Liang et al., 2023). In the Cyrillic injection attack, we perturbed the input text by replacing English characters with similar-looking Cyrillic characters. Specifically, we selected three highly frequent English vowels, "a", "e", and "o," and replaced them with their Cyrillic counterparts. For the paraphrasing attacks, we employed a PLM-based approach that incorporates the T5 model to paraphrase a given input text (Sadasivan et al., 2023).

4.5 Results and Discussion

This section discusses the experimental results under AI-generated news detection, including additional experiments on feature importance and PLM choice for the **J-Guard**. Furthermore, we empirically show the adversarial robustness of the **J-Guard** by emulating multiple attack scenarios.

4.5.1 RQ1 - AI-generated News Detection Performance

Here, we present an evaluation of the performance of AI news detection using a wide range of AI generators. Table 1 reports the AUROC scores for different detectors (rows) across different AI generators/PLMs (columns). Based on the results in Table 1, we make the following observations regarding AI-generated news detection:

1) **Effectiveness of journalism features** - when we look at the logistic regression results (1st 3 result rows of Table 1), we can see that journalism features outperform simple BOW and word2vec performance across all the AI generators. This suggests that the journalism feature space provides a reasonable boundary for discriminating between human-written news and AI-generated news.

2) **Effectiveness of J-Guard** - Our proposed method outperforms all the detection baselines in 4 out of 6 AI generators. However, for PPLM$_{gpt2}$ and GPT2 generators, we observe that the fine-tuned OpenAI detector (OpenAI$_{FT}$) outperforms **J-Guard** by a small margin. The OpenAI detector has an advantage in detecting GPT2 and PPLM$_{gpt2}$ as it is exposed to GPT2 samples in the first stage of finetuning done by OpenAI.

3) **Effectiveness of task-specific training** - We observe that off-the-shelf zero-shot methods (GLTR, DetectGPT, and OpenAI$_{Zero}$) perform poorly across many AI generators in detecting AI news. However, the performance improves significantly when we further finetune the OpenAI$_{Zero}$ on the AI news detection task (OpenAI$_{FT}$). This observation highlights the importance of task-specific supervision.

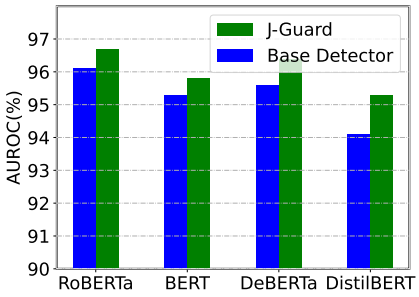

Figure 3: Effect of the choice of PLM for framework **J-Guard**- Average AUROC across all six AI generators, before and after Journalism guidance.

We also analyzed the impact of the base AI detector choice on our framework, **J-Guard**. We experimented with multiple open-source PLMs, as shown in Figure 3. We evaluate each PLM with and without **J-Guard** to evaluate the detection performance and report the average performance across the AI generators considered in our study. We found that the detection performance could be enhanced with the use of **J-Guard** on each PLM. Among all the models, RoBERTa yielded the best performance.

Dataset →	TuringBench					N.News	In-House Data	
Generator → / Detector ↓	Grover	CTRL	PPLM$_{gpt2}$	GPT2	GPT3	Grover	GPT3.5	GPT3.5$_{JG}$
LR+ BoW	0.816	0.775	0.792	0.822	0.806	0.896	0.810	0.805
LR+ W2V	0.854	0.793	0.804	0.871	0.852	0.915	0.847	0.838
LR + JF	0.897	0.831	0.873	0.931	0.912	0.933	0.883	0.847
GLTR	0.482	0.784	0.634	0.542	0.454	0.499	0.728	0.688
DetectGPT	0.549	0.806	0.492	0.505	0.557	0.815	0.766	0.751
OpenAI$_{Zero}$	0.746	0.763	0.918	0.857	0.773	0.962	0.756	0.718
OpenAI$_{FT}$	0.975*	0.969*	**0.966**	**0.980**	0.951*	0.993*	0.925*	0.911*
J-Guard	**0.986**	**0.972**	0.965*	0.975*	**0.968**	**0.998**	**0.934**	**0.917**

Table 1: Proposed **J-Guard** model performance (AUROC) values for AI-generated news detection. Bold shows the best AUROC within each column (Detector-PLM generator combination); asterisk (*) denotes the second-best AUROC.

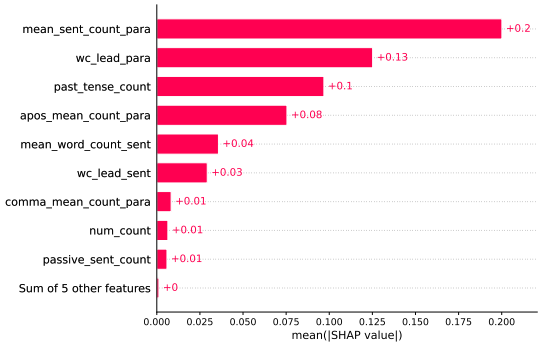

Figure 4: SHAP values to estimate journalism feature importance.

Additionally, we conducted a study to better understand the significance of journalism features in AI news detection with the help of a Shapley Additive Explanations (SHAP) (Lundberg and Lee, 2017) explainer on the logistic regression classifier. The SHAP values were used to indicate feature importance. We only present SHAP plots for the GPT3 detection task for brevity reasons, but SHAP plots related to other AI generator detection tasks can be found in the appendix section D.2. The SHAP plots show that certain features, such as mean sentence count for a paragraph (*mean_sent_count_para*), the word count of lead paragraph size (*wc_lead_para*), and past tense usage (*past_tense_count*) are highly significant in distinguishing AI news from human-written news, as depicted in Figure 4.

4.5.2 RQ2 - Adversarial Robustness of AI News Detection

This section discusses our experiments on evaluating the adversarial robustness of AI news detectors. As outlined in section 4.4, we conducted two types of attacks on the detectors: character-level attacks involving Cyrillic injection and word-level attacks involving paraphrasing. Table 2 shows the detectors' performance (AUROC) difference before and after each attack. For brevity, we only report the detection performance on GPT3 and ChatGPT data, while the other results can be found in the appendix section D.2. Based on the results presented in Table 2, we make the following observations regarding the adversarial robustness of AI news detectors:

Attack Success - We have observed that almost every SOTA baseline detector we have considered is susceptible to adversarial attacks. On average, the performance of the detectors dropped by at least 15-20%. In contrast, we observed a low attack success rate with the GLTR model. However, the observation of low attack success is meaningless as the GLTR model had a near-random guess (≈ 0.5 AUROC) performance before the attack. The overall reduction in performance following the Cyrillic injection attack can be attributed to the tokenizer. Cyrillic letters in the input text alter the token representation, subsequently affecting detection. For paraphrasing, modifying the original text could alter the learned decision boundary during detector training, leading to a performance decline.

Improved Adversarial Robustness of J-Guard - We have observed that **J-Guard** is quite resilient to adversarial attacks, with an average performance drop of only 7%. It is apparent that this robustness is due to the journalism features employed by **J-Guard**. For example, OpenAI$_{FT}$, which shares the same PLM architecture and training data for detection as **J-Guard**, has an average performance drop

Generator →	GPT3		ChatGPT	
Attack → / Detector ↓	Para.	Cyri.	Para.	Cyri.
GLTR	**0.041**	0.055	0.095	0.056
DetectGPT	0.222	0.196	0.254	0.183
OpenAI$_{Zero}$	0.244	0.201	0.223	0.154
OpenAI$_{FT}$	0.159	0.138	0.166	0.150
J-Guard	0.090	**0.041**	**0.091**	**0.040**

Table 2: Detector performance change after the attack (AUROC before the attack - AUROC after the attack). Bold shows the lowest AUROC difference within each column (detector-attack combination).

of nearly 15%. In the journalism feature space, we check for high-level semantic gaps and violations of journalism standards. Character-level attacks, such as Cyrillic injection, have a negligible effect on these feature calculations. Even with paraphrasing attacks, the edit distance between the original and perturbed text may be substantial in the input space but insignificant in the journalism feature space, making **J-Guard** robust to such attacks.

4.6 Further Analysis: Better Prompting versus J-Guard

To ensure consistency across data generations, we adhered to a headline-based prompt from the TuringBench dataset for our in-house GPT3.5 data production. However, given the capabilities of advanced LLMs like GPT 3.5, providing more contextual instructions is feasible. Thus, we explored the potential of prompting GPT 3.5 to adhere to real-world journalism standards in news article generation and its subsequent impact on the **J-Guard**'s detection abilities. For this evaluation, a test dataset comprising 1000 data points from ChatGPT was developed using what we term as a journalism-guided prompt (GPT3.5$_{JG}$) as shown below:

System prompt: "You are a helpful assistant. I want you to act as a journalist. Adhere to journalistic ethics, and deliver accurate reporting using your own distinct style."

User prompt: "Following the rules in the AP style guide for journalists, write a news article with the <headline>."

As presented in Table 1, there is a marginal decline in performance when using the journalism-guided prompt test dataset. This suggests that ChatGPT emulated the AP standards more effectively with this prompt than with the standard headline prompt. The performance of **J-Guard** clearly validates the strength of the proposed hypothesis, even when the prompt explicitly guides the LLM towards real-world journalistic generation. Yet, this analysis raises crucial questions for future research: Can we design a prompt that consistently directs LLMs to adhere to real-world journalism processes? And, if successful, can we detect such generated news articles?

5 Related Work

5.1 AI-generated Text Detection

Several methods have been explored for detecting AI-generated text, such as logistic regression, SVC, etc. (Ippolito et al., 2019). GLTR (Gehrmann et al., 2019) uses a set of simple statistical tests to check whether an input text sequence is AI-generated or not. Fine-tuned PLM detectors are also used and considered state of the art (Solaiman et al., 2019; Jawahar et al., 2020; Zellers et al., 2019; Kumarage et al., 2023), such as OpenAI's GPT2 detector that uses a RoBERTa backbone finetuned with GPT-2 outputs (Radford et al., 2019). With the rapid advancement of newer language models like GPT3.5/4, there is a growing emphasis on the capabilities of few-shot or zero-shot detection and the interpretability of these detectors (Mitrović et al., 2023). Some new detectors include commercial products such as GPTZero [5] and OpenAI's detector that is trained on the text generated by GPT-3[6]. An interesting zero-shot detection approach, DetectGPT (Mitchell et al., 2023), operates on the hypothesis that minor rewrites of AI-generated text would exhibit lower log-probabilities under the model compared to the original sample. Watermarking (Kirchenbauer et al., 2023) on PLM-generated text has gained attention as a detection mechanism in the research community. However, its success hinges on the cooperation and support of the organizations that develop the PLMs.

5.2 Adversarial Robustness of AI Text Detection

Multiple studies have examined the vulnerability of AI text detectors, specifically those designed for early PLMs like Grover and GPT2 (Crothers et al., 2022; Liang et al., 2023; Gagiano et al., 2021). These studies conducted various attacks at the character and word levels, including flipping

[5] https://gptzero.me/

[6] https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text

upper-lower case, using homoglyphs, misspelling words, and replacing synonyms. The results indicate that supervised-PLM-based AI text detectors are highly susceptible to these attacks, with success rates reaching up to 96% in some cases (Crothers et al., 2022). Recent research has also demonstrated that paraphrasing the input text can significantly undermine the performance of AI text detection approaches (Sadasivan et al., 2023; Krishna et al., 2023), raising concerns about the reliability of such methods. Proposed solutions involve semantic retrieval to counter paraphrase attack (Krishna et al., 2023), but they rely on text generation APIs like the OpenAI API, which limits their practical applicability when evaluating independent detection mechanisms.

Previous research has highlighted two important considerations for detecting AI-generated text. 1) it is impractical to rely on a single detector for all types of AI text, emphasizing the need for domain-specific models, and 2) ensuring the detector's robustness against adversarial attacks is critical, warranting further investigation in this field.

6 Conclusion

In this paper, we examine the task of detecting AI-generated news from a multidisciplinary perspective, aiming to identify domain-specific signals that can enhance detection accuracy while preserving robustness against adversarial attacks. We analyzed the real-world news production process compared to AI news generation and identified a set of stylistic cues that measure the deviation of AI-generated news from journalistic standards established by entities such as the Associated Press. Our proposed framework, **J-Guard**, incorporated these auxiliary features and steered existing supervised PLM-based AI text detectors to achieve robust performance across various text-generation AIs, including ChatGPT. For future work, it would be interesting to see how prompt engineering can generate news articles that evade journalism-guided detection.

7 Limitations

7.1 Assumption of Professional Journalism

In our study, we make the assumption that the human-written portion of the dataset is produced through a professional journalism process. This means that the news organization or journalist adheres to the journalism standards commonly defined by organizations such as the Associated Press (AP). It is important to note that our hypotheses and findings are valid only under this assumption. If the human-written articles come from a non-professional journalism source, we expect the detection performance to decrease since the distinction achieved through journalism features may no longer hold.

7.2 Domain-Specific Training

The approach we propose follows the supervised learning paradigm for AI news detection. As a result, it requires specific training data to be effective in real-world AI news detection scenarios. For instance, if we aim to ensure the performance of **J-Guard** on an AI text generator X, we first need to gather a training dataset consisting of news articles generated by X. It is important to emphasize that our approach does not claim to have cross-AI generator generalized detection capabilities. However, the set of journalism features we proposed are agnostic to the AI generator and derived from real-world journalism process analysis.

7.3 In-House Dataset

As described in our section 4.1, we generated our dataset using ChatGPT due to the lack of publicly available ones. Although we followed a similar data collection and generation pipeline as TuringBench (Uchendu et al., 2021), it is worth noting that there may be differences in the pre-processing and data cleanup we performed compared to the methods employed by the authors of TuringBench.

7.4 Generalizability for ChatGPT-generated Text Detection

Throughout our paper, we emphasize the specificity of our analysis and its focus on AI news detection. Therefore, the differences in ChatGPT text detection performance reported by the community [7], as opposed to the high-performance results presented in our work, can be attributed to the domain of the data, specifically news articles. We hypothesize that detecting a particular domain, such as news articles with a specific type and text style, is easier than detecting generic text generated by ChatGPT. In summary, our paper does not claim that **J-Guard** can be used for general ChatGPT text detection tasks; instead, it presents a specific method tailored

[7] https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text

to improve the detection of ChatGPT-generated news.

8 Ethical Considerations

8.1 Intended Use

It is crucial to consider the intended real-world application of **J-Guard** and its societal impact. Our research on AI news detection aims to develop an algorithm that effectively identifies and mitigates the spread of misinformative, AI-generated news articles. The primary application of our work lies in online content moderation and forensics, where the decisions made by our detector can be utilized to flag or remove news articles from social media platforms, web search results, and other platforms. However, a significant ethical concern arises from potential false positives generated by our method. Suppose the detector incorrectly flags a genuine news article from a reputable organization as AI-generated. In that case, it may lead to the censorship of legitimate news, causing harm to the reputation and rights of the journalist and the publishing organization. Hence, we strongly advise users not to incorporate **J-Guard** into fully automated real-world content moderation or forensics systems unless a human annotator or analyst works in conjunction with the system to make the final decision.

8.2 ChatGPT-generated News

In our study, we conducted experiments using the in-house ChatGPT-generated news articles. It is crucial to emphasize that we adhered to the usage policies[8] of OpenAI while generating these news articles through the API (refer to the prompt details in Section 4.1). We recognize the importance of not publicly releasing any AI-generated news article, as we cannot guarantee the factual accuracy of the content. Therefore, we will implement an on-demand release structure for our ChatGPT-generated news articles. Individuals or organizations requesting access to our generated news articles for legitimate academic research purposes will be granted permission to download the data.

8.3 Fairness and Bias in Detection

Our research endeavors to prioritize using natural language processing tools for the betterment of society while upholding principles of fairness and impartiality. We transparently disclose our methodology, results, and, most importantly, limitations to mitigate biases and address ethical concerns. Furthermore, we commit to continuous assessment and improvement of our system in the future.

8.4 Malicious Use of Adversarial Attacks

We understand the potential danger of an adversary misusing the adversarial attack setup we presented in our section 4.4 to attack existing commercial AI text detectors. However, we posit that finding these limitations and vulnerabilities in AI text detector systems (red-teaming) will outweigh the potential for misuse, given we help future researchers mitigate these issues. However, as a precaution, we will not release the adversarial setup code base to the public. Similar to ChatGPT data, individuals or organizations requesting access to our adversarial attack setup for legitimate academic research purposes will be granted permission to receive the code base.

9 Acknowledgement

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0123. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

[8] https://openai.com/policies/usage-policies

References

Marcel Broersma. 1880. Form, style and journalistic strategies. *An Introduction. W: M. Broersma (red.), Form and Style in Journalism. European Newspapers and the Representation of News*, 2005.

Evan Crothers, Nathalie Japkowicz, Herna Viktor, and Paula Branco. 2022. Adversarial robustness of neural-statistical features in detection of generative transformers. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Rinaldo Gagiano, Maria Myung-Hee Kim, Xiuzhen Jenny Zhang, and Jennifer Biggs. 2021. Robustness analysis of grover for machine-generated news detection. In *Proceedings of the The 19th Annual Workshop of the Australasian Language Technology Association*, pages 119–127.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*.

Josh A. Goldstein, Girish Sastry, Micah Musser, Renee DiResta, Matthew Gentzel, and Katerina Sedova. 2023. Generative language models and automated influence operations: Emerging threats and potential mitigations.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2019. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. 2020. Automatic detection of machine generated text: A critical survey. *arXiv preprint arXiv:2011.01314*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*.

Tharindu Kumarage, Joshua Garland, Amrita Bhattacharjee, Kirill Trapeznikov, Scott Ruston, and Huan Liu. 2023. Stylometric detection of ai-generated text in twitter timelines. *arXiv preprint arXiv:2303.03697*.

Gongbo Liang, Jesus Guerrero, and Izzat Alsmadi. 2023. Mutation-based adversarial attacks on neural text detectors. *arXiv preprint arXiv:2302.05794*.

Tommy Carl-Gustav Linden. 2017. Algorithms for journalism: The future of news work. *The journal of media innovations*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Samuel Mateus. 2018. Journalism as a field of discursive production–performativity, form and style. *Catalan Journal of Communication & Cultural Studies*, 10(1):63–77.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*.

Sandra Mitrović, Davide Andreoletti, and Omran Ayoub. 2023. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *arXiv preprint arXiv:2301.13852*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Ivor Shapiro. 2010. Evaluating journalism: Towards an assessment framework for the practice of journalism. *Journalism Practice*, 4(2):143–162.

Kai Shu, Amrita Bhattacharjee, Faisal Alatawi, Tahora H Nazer, Kaize Ding, Mansooreh Karami, and Huan Liu. 2020. Combating disinformation in a social media age. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1385.

Kai Shu, Suhang Wang, Thai Le, Dongwon Lee, and Huan Liu. 2018. Deep headline generation for clickbait detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 467–476. IEEE.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Reuben Tan, Bryan A Plummer, and Kate Saenko. 2020. Detecting cross-modal inconsistency to defend against neural fake news. *arXiv preprint arXiv:2009.07698*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon Lee. 2021. Turingbench: A benchmark environment for turing test in the age of neural text generation. *arXiv preprint arXiv:2109.13296*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Feature Extractors

In Section 3.2, we divided the feature extractor function set F into three subsets: $F^i|i \in 1, 2, 3$, where $F = F^1 \cup F^2 \cup F^3$. In this appendix section, we provide implementation details for the feature extractor functions in the categories mentioned above.

A.0.1 F^1: Organization and Grammar Standards

We used two NLTK tokenizers for our implementations: word_tokenize for word-level tokenization and sent_tokenize for sentence-level tokenization.

Mean Word Count (WC): The average word count per sentence in the input news article. First, we extracted the sentences using sent_tokenize, and then for each sentence, we obtained the word tokens. The word count for each sentence was determined by counting all word tokens that contained at least one alphabetical character.

Mean Sentence Count (SC): We counted the number of sentences per paragraph in the news article. To obtain the paragraphs, we split the input article using the newline character. For each paragraph, we obtained sentence-level tokens. The sentence count for each paragraph was determined by counting all sentences that contained at least one alphabetical character.

Word Count of the Leading Sentence: We focus on the lead sentence of the input news article. After obtaining the sentence-level tokens using sent_tokenize, we calculated the word count for this leading sentence using the same approach as the Mean Word Count (WC).

Word Count of the Leading Paragraph: Similar to the previous approach, we extracted all paragraphs by splitting the input article based on the newline character. We then counted the number of word tokens in the first paragraph.

Mean Sentence Count with Passive Voice: Here we determined the number of sentences in a paragraph written in the passive voice. To identify the voice of a sentence, we employed a simple test. First, we extracted the dependency tree relations using the spaCy dependency parser[9]. Then, we classified a sentence as passive if it contained either an 'agent' relation or an 'nsubjpass' relation.

Mean Sentence Count with Past Tense: The number of sentences in a given paragraph that were written in the past tense. After extracting the sentences for a given paragraph, we used POS tags to determine whether a sentence was in the past tense. Specifically, we checked if the sentence's POS tags included "VBD" or "VBN" and classified the sentence as past tense accordingly.

A.1 F^2: Punctuation usage

Under punctuation usage, we calculated the average occurrence of "!" "#", """, and the Oxford comma per paragraph. We divided the input news article into paragraphs, following the same approach as the previous feature extractions. For each paragraph, we determined the frequency of the mentioned punctuation.

A.2 F^3: Formatting standard violations

Here, we identify and tally violations related to date, time, and number formats in accordance with the AP standards.

Date format violations: To detect date format violations, we utilize the datefinder library[10] to extract date elements from the input news article. We then verify if the date adheres to the standard format specified by AP standards. Specifically, we ensure that the day is written in full without abbreviations (e.g., "Monday," "Tuesday," etc.), and that the month is represented correctly. For instance, when a month is used with a specific date, we expect abbreviations like "Jan.," "Feb.," "Aug.," "Sept.," "Oct.," "Nov.," and "Dec." When a phrase only lists the month and year, the month should be spelled out, and there should be no comma separating the month and year.

Time format violations: In assessing time format violations, we verify if time phrases in the news article adhere to the AP standards. This entails using lowercase "a.m." and "p.m." with periods and ensuring that numerals precede the time. If a time phrase does not conform to these standards, it is considered a format violation. Similar to the date format analysis, we employ the datefinder library to extract time phrases from the text.

Number format violations: We identify number format violations when numbers in an article fail to comply with the AP standards. According to these standards, numbers from zero to nine should be spelled out, while numerals should be used for 10 and above.

[9] https://spacy.io/api/dependencyparser

[10] https://pypi.org/project/datefinder/

Altogether we collected **14** journalism features through the above extractors.

B Baselines

GLTR (Gehrmann et al., 2019): This approach utilizes a proxy language model (PLM) to calculate the log probabilities of tokens in the input text. The authors then incorporate a set of statistical scores to predict the label, including average log probability, average token rank, token log-rank, and predictive entropy. For example, a higher average log probability of input indicates AI generation. The second and third scores share a similar assumption, where lower average ranks in input suggest AI-generated text. The final score is based on the hypothesis that AI-generated text tends to have lower entropy. Our paper reports the average performance across all the trials based on the scores mentioned above.

DetectGPT (Mitchell et al., 2023): This approach also utilizes a proxy PLM to calculate log probabilities for individual tokens. However, its decision process involves comparing the log probability of the original input text with the log probability of a set of perturbed versions of the input text. These perturbations are generated using the T5-base. The authors hypothesize that if the difference in log probabilities between the original text and the perturbed text is consistently positive, then it is likely that an AI model generated the input text.

OpenAI-GPT2 detector: This detector is a RoBERTa model fine-tuned on the GPT-2-output dataset[11] which consists of 250K documents from the WebText dataset (Radford et al., 2019) and 500K GPT2 generated data. We incorporate two variants of this model: 1) OpenAI$_{Zero}$ - off-the-shelf model without any additional finetuning on AI-generated news detection task and 2) OpenAI$_{FT}$ - off-the-shelf model further finetuned on training datasets used for AI generated news detection task.

C Implementation Details

Apart from the implementation details discussed in Section 4.3, another crucial aspect to consider is the hyperparameters of the Guidance Head and Classification Head, including layer sizes.

Guidance Head: As described in the methodology section, the Guidance Head comprises one hidden layer that maps the input $[B_d^{CLS}, J_n]$ to a higher-dimensional feature space. Consequently, we opted for a larger hidden layer size compared to the input size $d + n$ (Base detector hidden size + journalism feature vector size). In the case of RoBERTa-base $d = 768$, where $d + n = 768 + 14 = 782$, we found that a Guidance Head layer size of 1024 yielded the best performance. Additionally, we used an output layer of size 256 for optimal results. To summarize, the layers of the Guidance Head are structured as $782 \rightarrow 1024 \rightarrow 256$.

Classification Head: The decision regarding the layer size for the Classification Head was straightforward. Essentially, starting from the output size of the Guidance Head, our objective was to obtain the final prediction for the two classes. The layer sizes that achieved the best performance were $256 \rightarrow 32 \rightarrow 2$.

D Additional Experiment Results

D.1 Journalism Feature Importance

We conducted a study to better understand the significance of journalism features in AI news detection with the help of a Shapley Additive Explanations (SHAP) (Lundberg and Lee, 2017) explainer on the logistic regression classifier. In section 4.5, we only presented the SHAP plots for the GPT3 detection task for brevity reasons. Therefore, here we present the additional SHAP plots related to other PLMs detection tasks.

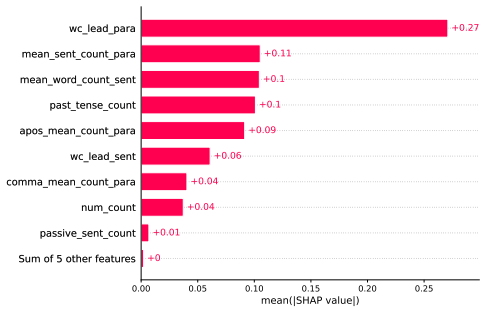

Figure 5: SHAP values to estimate journalism features importance - CTRL

We see that almost all the SHAP plots agree on the importance of certain features, such as leading paragraph and or sentence word count, apostrophe usage, and past tense usage.

D.2 Adversarial Robustness Results

In section 4.5.2, we discussed our experiments on evaluating the adversarial robustness of AI news detectors. We conducted two types of attacks on the

[11] https://github.com/openai/gpt-2-output-dataset

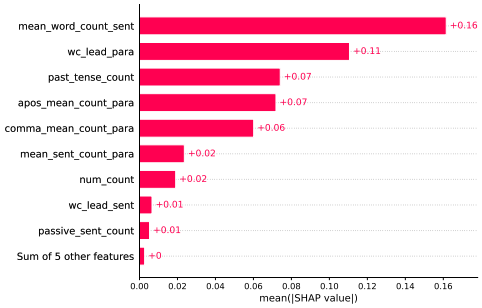

Figure 6: SHAP values to estimate journalism features importance - GROVER

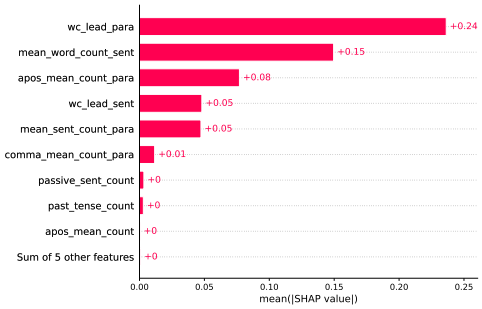

Figure 7: SHAP values to estimate journalism features importance - GPT2

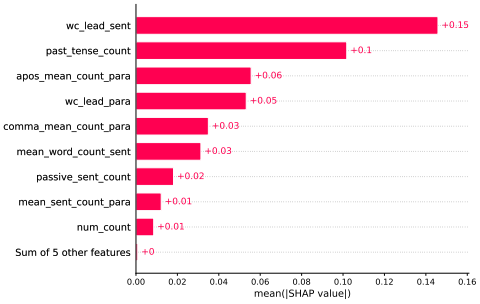

Figure 8: SHAP values to estimate journalism features importance - PPLM$_{gpt2}$

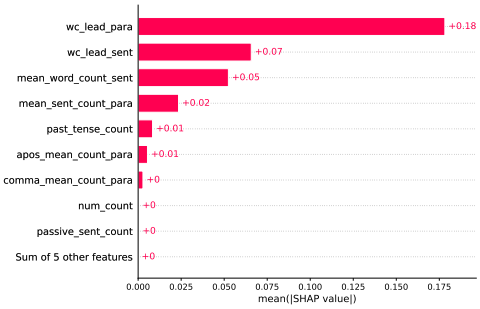

Figure 9: SHAP values to estimate journalism features importance - ChatGPT

detectors: character-level attacks involving Cyrillic injection and word-level attacks involving paraphrasing. We only report the detection performance on GPT3 and ChatGPT data for brevity in section 4.5.2. Therefore, here we present the results of the rest of the PLMs of our study. Table 3 and Table 4 show the detectors' performance (AUROC) difference before and after each attack.

Table 3 and Table 4 hold similar observations as we presented with GPT3 and ChatGPT3 data in section 4.5.2. Almost every SOTA baseline detector we have considered is susceptible to adversarial attacks. On average, the performance of the detectors dropped by at least 10-20%. However, in some cases, we observed a low attack success rate with the GLTR and DetectGPT. However, the observation of low attack success is meaningless as these models had a near-random guess (≈ 0.5 AUROC) performance before the attack. Similar to GPT3 and ChatGPT detection, we can observe that **J-Guard** is quite resilient to adversarial attacks across other PLM generators, with an average performance drop of only 7%. It is again evident that this robustness is due to the journalism features employed by **J-Guard**. For example, OpenAI$_{\text{FT}}$, which shares the same PLM architecture and training data for detection as **J-Guard**, has an average performance drop of nearly 15%.

Table 3: Detector performance change after the attack (AUROC before the attack - AUROC after the attack).

Generator →	Grover		CTRL	
Attack → / Detector ↓	Para.	Cyri.	Para.	Cyri.
GLTR	**0.035**	**0.038**	0.233	0.186
DetectGPT	0.127	0.086	0.248	0.195
OpenAI$_{Zero}$	0.233	0.169	0.229	0.175
OpenAI$_{FT}$	0.144	0.113	0.162	0.106
J-Guard	0.082	0.054	**0.074**	**0.031**

Table 4: Detector performance change after the attack (AUROC before the attack - AUROC after the attack).

Generator →	PPLM$_{gpt2}$		GPT2	
Attack → / Detector ↓	Para.	Cyri.	Para.	Cyri.
GLTR	0.124	0.082	0.090	0.054
DetectGPT	0.083	0.050	0.083	0.029
OpenAI$_{Zero}$	0.187	0.123	0.237	0.170
OpenAI$_{FT}$	0.140	0.092	0.181	0.137
J-Guard	**0.082**	**0.042**	**0.073**	**0.040**