Deepfake Text Detection: Limitations and Opportunities

Jiameng Pu*¶, Zain Sarwar†¶, Sifat Muhammad Abdullah*, Abdullah Rehman*, Yoonjin Kim*,
Parantapa Bhattacharya§, Mobin Javed‡, Bimal Viswanath*
*Virginia Tech, †University of Chicago, ‡LUMS Pakistan, §University of Virginia
*{jmpu, sifat, abdullahzr, ykim05, vbimal}@vt.edu, †zsarwar@uchicago.edu, §parantapa@virginia.edu,
‡mobin.javed@lums.edu.pk

***Abstract*—Recent advances in generative models for language have enabled the creation of convincing synthetic text or deepfake text. Prior work has demonstrated the potential for misuse of deepfake text to mislead content consumers. Therefore, deepfake text detection, the task of discriminating between human and machine-generated text, is becoming increasingly critical. Several defenses have been proposed for deepfake text detection. However, we lack a thorough understanding of their real-world applicability. In this paper, we collect deepfake text from 4 online services powered by Transformer-based tools to evaluate the generalization ability of the defenses on content in the wild. We develop several low-cost adversarial attacks, and investigate the robustness of existing defenses against an adaptive attacker. We find that many defenses show significant degradation in performance under our evaluation scenarios compared to their original claimed performance. Our evaluation shows that tapping into the semantic information in the text content is a promising approach for improving the robustness and generalization performance of deepfake text detection schemes.**

***Index Terms*—deepfake text, deepfake detection**

I. Introduction

Progress in natural language generation (NLG) has enabled deep learning models such as GPT-2 [1] and GPT-3 [2] to generate *synthetic text* or *deepfake text* with high linguistic quality. Both models fall in the Transformer [3] family of language models (LMs). These large LMs with billions of parameters, can generate synthetic text on diverse topics. They have many applications, including, generating content for entertainment purposes (*e.g.,* stories, jokes) [4], dialog systems [5], text summarization [6], and automated journalism [7].

Unfortunately, such technology raises serious security concerns—synthetic text can be misused to power several threats aimed at misleading content consumers. Zellers *et al.* demonstrated that GPT-2-based LMs can generate convincing fake news articles [8]. Such tools can enable large-scale disinformation campaigns. Yao *et al.* [9] showed that synthetic text can be used to create fake online restaurant reviews. Such threats will lead to users losing trust in online content, including crowd-sourced information. Other serious threats include automated email generation for targeted attacks [10], and synthetic text that can radicalize individuals into having violent, extremist ideologies [11].

One approach to curb the misuse of synthetic text is to have humans evaluate and flag the text. However, such an approach is unlikely to scale, and more importantly, existing works already demonstrate that humans are unable to reliably distinguish between real and synthetic text. Recent work showed that the GPT-3 [2] model can produce realistic news articles, which humans identify as synthetic only 52% of the time. Therefore, it is pertinent that we develop robust automated schemes to accurately detect synthetic text.

Fortunately, the research community has developed several defenses to automatically detect synthetic text [8], [9], [12]–[14]. They are all supervised learning schemes that use a language model to extract features to build a classifier. Some defenses only focus on features that capture statistical artifacts or imperfections in the generated text [12]. Our reproduction of 6 of the best defenses show that they all achieve high detection performance, ranging from 79.6% F1 score to 98.5% F1 score in detecting synthetic samples. However, we lack a thorough understanding of the real-world applicability of these defenses.

To understand real-world applicability, we focus on two key aspects: (1) *How well would existing defenses perform when applied to synthetic text in the wild?* All existing defenses have only been tested on synthetic datasets produced by the research community themselves. It is unclear how well they would work in the real world. To make matters more challenging, there is a lack of real-world synthetic datasets that the research community can use to study such defenses. (2) *How would existing defenses perform against an adaptive attacker who is knowledgeable about the defenses?* Existing defense work provides limited understanding of threats posed by adaptive attackers. Of course, the community is well aware of attacks that craft adversarial samples to fool text classifiers [15]. Such attacks are also feasible against synthetic text detection classifiers. However, such adversarial attacks may not always be practical. For example, text adversarial attack schemes like TextFooler assume a black-box scenario that requires a large number of queries to the victim defense model to craft adversarial samples. Such query-based attacks could be caught by looking for specific query patterns [16]–[18]. Instead, we argue that there are much simpler, low-cost (computationally), yet effective strategies to fool existing defenses. To address these research questions, we conduct a study of synthetic text detection schemes through the lens of security. Our

[¶] These authors contributed equally to this work.

contributions are as follows:

- We conduct a measurement study to collect synthetic text in the wild, and introduce 4 new real-world synthetic datasets. This includes synthetic data obtained from 3 *text-generation-as-a-service* platforms (geared towards the SEO community), and synthetic data produced by a GPT-3 powered bot on the web (on Reddit.com). We find that Internet users and services are already using state-of-the-art Transformer-based models to synthesize text.
- We evaluate performance of 6 state-of-the-art defenses on our new real-world synthetic datasets. Many defenses show significant degradation in performance compared to their original claimed/reproduced performance. Defenses using entity-based semantic features and those using robust pre-training methods combined with bidirectional context, generalize better to content in the wild.
- We develop simple, computationally low-cost attacks that modify the attacker's text generation process to evade detection. Our experiments show that just changing the decoding or text sampling strategy is sufficient to break many defenses. Moreover, defenses that are trained to look for specific text decoding artifacts are easier to evade by changing the text decoding strategy.
- We propose and evaluate a new black-box adversarial sample crafting strategy called DFTFooler. Our attack requires no queries to the defense model, and exploits insights unique to the synthetic text detection problem. DFTFooler only requires a publicly available pre-trained language model to craft adversarial perturbations. DFTFooler can produce transferable adversarial samples that can degrade the performance of multiple defenses.
- Lastly, our analysis shows that a promising approach to improve defenses is to tap into the semantic information in the text content. Our analysis of the existing defense called FAST [14] shows that using entity-based features capturing the factual structure of the text can lead to better adversarial robustness and generalization performance.

Our study provides many actionable insights that can be used to improve real-world applicability of defenses. Datasets and code used in this study are available on *GitHub*.[1]

II. Background and Goals

In this work, *synthetic text* refers to text generated by DNN-based LMs, and *real text* refers to human-written text.

Using language models for text generation. Synthetic text is generated using an LM. An LM is a statistical model that provides the joint probability distribution for a sequence of n tokens $x_1, \cdots, x_n$. Tokens can be characters, words, or subword tokens [19]. This joint distribution can be factorized by computing the conditional probability for each token, given the previous tokens:

$$p(x_0, \cdots, x_n) = \prod_{t=0}^{n} p(x_{t+1} \mid x_0, \cdots, x_t) \tag{1}$$

Given a LM that provides the conditional probability in Equation 1, synthetic text is generated using the following steps: Feed an initial *priming sequence*, which can be a single token, x_0, into the LM, which then provides the conditional probability for the next token as $p(x_1 \mid x_0)$. This priming sequence can also be a special "start-of-text" token (which the model is trained to recognize) or a sequence of tokens. In the second step, the next likely token, x_1, is sampled from the distribution over the token vocabulary—a process known as *decoding*. Next, we can generate the next token, by feeding (x_0, x_1) back into the LM, and sampling x_2, using the same decoding strategy. By repeating this process, one can generate text until a desired sequence length is reached, or an "end-of-text" token is chosen (which the model is trained to produce).

Two key factors impact the quality of synthetic text—the decoding function, and the LM architecture:

Text decoding strategies. Decoding strategies have witnessed significant development in recent years, and are now capable of producing high-quality synthetic text. Two effective decoding strategies are Top-k sampling [20] and Top-p or Nucleus sampling [21]. In Top-k sampling, the distribution is truncated and re-normalized to keep the k most probable tokens, and then a token is randomly sampled. Holtzman et al. [21] developed Top-p (or Nucleus) sampling, which produces more diverse and high quality text than Top-k sampling. In Top-p sampling, one truncates the distribution to keep the most probable tokens, such that their cumulative probabilities are greater than or equal to p, where $p \in [0, 1]$. Another approach, Temperature sampling [22], shapes the probability distribution by dividing the logits by a temperature parameter before passing them to the softmax function. Low temperatures make the model produce less diverse text, as it makes more confident predictions (tokens), while temperatures greater than 1, result in more diverse text, as confidence is decreased. The simplest decoding strategy is greedy sampling, where the most probable next token is chosen. However, temperature sampling and greedy sampling tend to produce repetitive text [21]. Other sampling strategies such as beam search sampling [23] are also known to suffer from similar problems [21].

DNN-based LM architectures. Until 2017, the de facto choices were RNNs [24] and LSTMs [25]. These models use a recurrent loop to maintain an internal "hidden state" that stores information about previous tokens, which is used to compute the next token distribution. However, they have limitations, including vanishing/exploding gradients [26], the sequential nature of the model limiting parallelization, and an inability to generate longer coherent text [27].

In 2017, Vaswani et al. proposed the Transformer [3] architecture to address limitations of RNNs. Transformers, in contrast with RNNs/LSTMs, are based on the "attention" mechanism [28]. Instead of using a single hidden state to represent all previous tokens, attention mechanisms allow the model to compute vector representations of each token separately. These representations take into account context and relationship with all other tokens. Attention mechanisms

[1] https://github.com/jmpu/DeepfakeTextDetection

can be customized to "pay attention" to only previous tokens (unidirectional attention), or to pay attention to future tokens, assuming they are provided (bidirectional attention). While the bidirectional attention is not suitable for text generation (future tokens are not available when generating text), it is useful when computing representations for other NLP tasks, *e.g.,* classification. Another advantage of Transformers is that each token can be processed in parallel, and is not dependent on previous tokens. This is made possible by using the teacher-forcing paradigm [29]. Transformers now power state-of-the-art for many NLP tasks. Examples include the popular BERT [30], RoBERTa [31], GPT-2 [1], and GPT-3 [2] models.

Goals. Our goal is to understand the real-world applicability of existing defenses to detect synthetic text. We focus on two directions: (1) *Understanding and improving performance of defenses in the wild.* The research community has made significant progress in developing detection schemes [8], [9], [12]–[14]. However, these defenses have been primarily tested on synthetic text produced by researchers themselves. It is unclear how well these methods would generalize to synthetic text in the wild, *i.e.,* those produced by the Internet community. We collect real-world synthetic text to understand the performance of existing detection schemes. We also propose efficient methods that adapt existing defenses to improve performance in the wild. (2) *Understanding and improving performance against adaptive attackers.* Before deployment, defenders should consider an adaptive attacker who is knowledgeable about the defense and aims to evade detection. Existing works on evasion strategies primarily focus on generic black-box attacks that require a large number of queries to the defender's model [32]. However, the defender's model may not even be exposed as a public API for queries. Also, one can detect query-based adversarial sample crafting schemes by looking for specific query patterns [16], [17]. Using a surrogate model and relying on the adversarial samples to "transfer" may not always be effective either [33], [34]. Instead, we investigate more practical, (computationally) low-cost evasion strategies that require no queries to the defender's model, and no surrogate model as well. We also investigate methods to improve resilience against the proposed evasion strategies.

III. Models, Datasets and Metrics

We study 6 state-of-the-art defenses for detecting synthetic text. To study synthetic text in the wild, we introduce 4 new synthetic datasets. We present metrics to evaluate defense performance on real-world datasets and against adaptive attackers.

A. Defenses: Synthetic Text Detection Schemes

Existing defenses are supervised learning schemes that use a LM to extract features to build a binary classifier (real vs. synthetic). We consider 5 existing defenses, including GROVER [8], GLTR-BERT [12], GLTR-GPT2 [12], BERT-Defense [13], and FAST [14]. Inspired by BERT-Defense, we build an additional defense, called RoBERTa-Defense.

Defense performance metrics. Existing defenses are evaluated using a variety of metrics. BERT-Defense reports accuracy and AUC, whereas GLTR reports only AUC. FAST and GROVER use a modified version of accuracy, called paired accuracy (in addition to normal accuracy). Paired accuracy is computed by pairing real and synthetic articles such that both articles share the same metadata, *e.g.,* article title. If the detector assigns the synthetic article a higher probability than the real class, it is considered to be correctly classified. However, the paired accuracy setting is unrealistic because it assumes access to real articles used to generate the synthetic articles. Therefore, we do not use it in our study. We report all results primarily using the F1 score, Precision, and Recall for the synthetic class. These metrics provide insights into class-specific detection performance (unlike accuracy) and does not delegate the task of calibrating a decision threshold to future work. Additionally, we report AUC ROC scores for the 6 defenses on their original test datasets in Table I.

Defenses. Table I provides an overview of the training setup for each defense, and their performance. GROVER, RoBERTa-Defense, and FAST are trained to detect synthetic news articles, while GLTR-BERT, GLTR-GPT2, and BERT-Defense are "open-domain" schemes and applicable to diverse topic domains.

(1) GROVER. Proposed by Zellers *et al.* [8], it is a framework that can both generate and detect synthetic news articles. They first train a synthetic news article generator using a GPT-2 based LM. This generator is trained on the RealNews dataset [8], a large corpus (120GB) of news articles from Common Crawl [35]. The GPT-2 model is modified to incorporate context fields for relevant metadata, *e.g.,* date, author names, article title, web domain and body text. This generator can produce synthetic news articles conditioned on the metadata. Next, to build a detection scheme, a classification layer is attached to extract information from the hidden state of the special [CLS] token (placed at the end of each article). This updated model is fine-tuned on synthetic articles generated by GROVER and more real articles from the RealNews dataset. GROVER performs well when detecting text generated by GROVER itself. However, this is not a realistic setting, as the defender is unlikely to have access to the attacker's generator. We study GROVER in more realistic settings.

Our GROVER setup: We use the largest, publicly released version (1.5B parameters) of the GROVER classifier called GROVER-Mega [36]. Details of the training and testing setup are in Table I. GROVER is fine-tuned using 5000 real articles from the RealNews dataset, and 5000 synthetic articles generated by GROVER itself. Using this publicly available model, we obtain an F1 score of 87.1% on their original test set [8].

(2) GLTR-BERT and GLTR-GPT2. GLTR [12] uses the insight that decoding strategies tend to sample tokens that are assigned high probabilities by the LM. Hence, synthetic text can be detected by analyzing the likelihood of tokens in the text sequence, as determined by a LM. Presence of many high probability tokens is an indication that the text sample is likely synthetic. Using a LM, GLTR extracts features based on the number of tokens in the Top-10, Top-100, and Top-1000 ranks as determined by the token probability distributions. The

Defense Models	Train & Test Datasets		Decoding Setting	Train Set Size/Class	Test Set Size/Class	Performance (%)			
	Real	Synthetic				F1	P	R	AUC
BERT-Defense	WebText [1]	GPT2-Large [1]	Top-p 0.96	10,000	4,000	88.8	91.7	86.0	95.9
GLTR-GPT2	WebText	GPT2-XL [1]	Top-k 40 Temp 0.7	4,000	4,000	98.5	98.9	98.1	99.8
GLTR-BERT	WebText	GPT2-XL	Top-k 40 Temp 0.7	4,000	4,000	79.6	78.7	80.6	86.5
GROVER	RealNews [8]	GROVER [8]	Top-p 0.94	5,000	4,000	87.1	83.4	91.1	94.3
FAST	RealNews	GROVER	Top-p 0.96	5,000	4,000	87.0	83.8	90.4	93.7
RoBERTa-Defense	RealNews	GROVER	Top-p 0.96	5,000	4,000	86.3	81.6	91.7	93.9

TABLE I: Details regarding the training and evaluation of the defenses. From left to right: Datasets used for training and testing defense models; Decoding strategy used to generate synthetic data; Number of samples in training and test datasets; Detection performance of the defenses (F1, P, R, AUC represents F1 score, Precision, Recall, AUC ROC score, respectively).

features are then fed to a Logistic Regression classifier.

Our GLTR setup: We use the authors' code to extract the features [37], but no code for building the Logistic Regression classifier was released. The authors reported using Logistic Regression with default settings in the scikit-learn library [38]. We additionally apply grid search on the hyperparameters to ensure the classifier is properly tuned (See Appendix VII-E for details). To build an open-domain classifier, we train on synthetic text from GPT2-XL (similar to the original work) and real articles from the WebText dataset [1]. Both sources are known to cover diverse topics. Similar to the original work [12], we create 2 variants of GLTR, namely GLTR-BERT that uses BERT [30], and GLTR-GPT2 that uses the GPT2-XL [1] as the back-end LM.[2] We obtain F1-scores of 98.5% and 79.6% for GLTR-GPT2 and GLTR-BERT, respectively.

(3) BERT-Defense. A BERT-based binary classifier, proposed by Ippolito *et al.* [13], attaches a classification layer to a pre-trained BERT-Large LM, and then fine-tunes it on a dataset of synthetic and real articles.

Our BERT-Defense setup: The authors did not release the datasets and models. We replicated their experimental setup. See Table I for details. While the original work reported an accuracy of 81%, our model achieves an F1 score of 88.8% on the test set. We achieve a higher F1 score, even after using a smaller training set (10,000 articles per class versus 250,000 articles in the original work). Our model uses the full context window size of 512 tokens, supported by BERT, unlike the smaller window size of 192 tokens in the original. We suspect the higher performance is likely due to the larger context window used by our implementation.

(4) RoBERTa-Defense. Inspired by BERT-Defense, we create an additional defense using the same approach, but with a different language model, RoBERTa [31]. RoBERTa makes several changes to the BERT LM, such as training the model on a larger dataset with a bigger batch size, removing the next-sentence-prediction task, training on longer sequences and dynamically changing the masking pattern applied to the training data. RoBERTa is known to outperform BERT on NLP tasks such as GLUE [39], SQuAD [40], and RACE [41].

Our RoBERTa-Defense setup: We train a RoBERTa-base model on synthetic text produced by GROVER and real text obtained from the RealNews dataset, and obtain an F1 score of 86.3% in detecting synthetic news articles.

(5) FAST. FAST, proposed by Zhong *et al.* [14], unlike other defenses, taps into the semantic layer or the "factual structure" of text. FAST uses a graph-based learning approach that uses features based on named entities in the document. FAST exploits the insight that state-of-the-art text generators still produce inconsistencies in the factual structure of text, when compared to real text. For example, while it is easy for humans to correctly mention and reference named entities (*e.g.,* location, people, objects) across sentences, a text generator might make mistakes and create inconsistencies in how entities are mentioned in continuous sentences. To capture such inconsistencies, FAST constructs an entity network based on how entities are referenced within and across sentences, and uses a graph convolution network (GCN) to learn patterns in the network. In addition, FAST also uses the RoBERTa LM [31] to extract token, and document-level representations, in conjunction with the GCN-based features. We analyze FAST in detail in Section V-C.

Our FAST setup: A pre-trained model for FAST was not available. We obtained code for FAST, and reproduced the experimental setup described in the original work. Similar to the original work, we train FAST to detect synthetic news articles, and train it on the RealNews dataset (real class) and text from GROVER (synthetic class). [3] We obtain an F1 score of 87% in detecting news articles (Table I).

Context window size for training and evaluation. All 6 defenses use a context window size of the first 512 tokens in an article for detection, as determined by its tokenization scheme. To understand the impact of the context window size for detection, we evaluate all defenses against smaller

[2] GLTR [12] used GPT2-small as the LM, but we use a larger and more accurate LM, GPT2-XL.

[3] Zhong *et al.* [14] additionally trained a version of FAST on WebText (real) and GPT2-XL text (synthetic). We omitted this setting for simplicity.

context window sizes, *i.e.,* 64, 128, 256 tokens, and present results in Figure 5 in the Appendix. Performance (F1 score) monotonically increases as the context window size increases. Therefore, we chose a 512 token window size. Larger window size would significantly increase computational complexity for all experiments, and some pre-trained models only support a certain maximum window size (*e.g.,* 512 tokens for BERT).

Other defenses. There are a few other defenses that are not considered in our study. For example, Yao *et al.* [9] proposed a method to detect LSTM-generated synthetic reviews. We omit it because our preliminary evaluation yielded unsatisfactory results—an F1 score of only 68% in detecting GROVER generated text. We discuss more details of these experiments and other defenses in the Appendix VII-D.

B. Evaluation Metrics

We use the following metrics to measure defense performance on text in the wild and against adaptive attackers:

Percentage change in detection performance: ΔF1, and ΔRecall. We measure the percentage change in detection performance for the synthetic class, when applied to a new test dataset, compared to a specific baseline performance. This is broken into percentage changes in F1 and Recall, *e.g.,* $\Delta F1 = (F1_{new} - F1_{baseline})/(F1_{baseline})$. The new test dataset can be an *In-the-wild* dataset, or a dataset containing synthetic text produced by an adaptive attacker. We define baseline performance depending on the experiment context, which usually refers to the defense performance when evaluated on the test datasets considered in the original work (*e.g.,* numbers in Table I). For attack experiments, where only synthetic samples are modified in the test set (*i.e.,* real samples are the same in the test and baseline settings), we only consider Δ Recall (for the synthetic class), as there will be no change in false positives (real samples classified as synthetic). Note that the change in performance can be a degradation or improvement in performance.

Evasion rate (ER). In some attacks, the adaptive attacker perturbs existing synthetic samples to evade detection. In such settings, we use evasion rate, defined as the fraction of perturbed synthetic samples that evade detection by a defense. Higher fraction indicates higher attack success.

Evaluating quality of synthetic text. For adaptive attackers, it is not sufficient to evade detection, it is also necessary to maintain high linguistic quality of the generated text. We measure linguistic quality using the state-of-the-art GRUEN metric [42]. Zhu *et al.* proposed GRUEN, an unsupervised, reference-less metric designed for synthetic text. GRUEN correlates highly with human judgements, and better than other existing metrics for linguistic quality. The metric, computed for a synthetic sample, ranges from 0 to 1, and a higher value indicates better linguistic quality. An advantage is that this metric does not require any reference text, which usually requires human effort to obtain. GRUEN measures linguistic quality based on "grammaticality", non-redundancy, discourse focus, structure and coherence.[4] More details are in the Appendix VII-B. Other linguistic quality metrics have also been proposed over the years. In the Appendix VII-A we describe other metrics, and justify our choice of using GRUEN over the other metrics.

Datasets	**#Documents per Class**	**Document Topic(s)**
AI-Writer	1,000	News
ArticleForge	1,000	News
Kafkai	1,000	Cyber Security, SEO, Marketing
RedditBot	887	Reddit Comments

TABLE II: Details of the *In-the-wild* datasets.

C. In-the-wild Datasets

We collect 4 *In-the-wild* datasets from the web containing both synthetic and real articles from matching semantic categories. All measurements were conducted from Nov. 2020 to Apr. 2021. This includes synthetic text posted by Internet users, and text from synthetic *text-generation-as-a-service* platforms, geared towards the SEO community. Synthetic text generation services could be misused to create fake news articles, fake reviews, or fake web articles for BlackHat SEO activities. While we could not verify the text generators used by the services we study, they claim to use customized versions of Transformer-based LMs. This again highlights the need to understand real-world performance of defenses, because text generators used in the wild can be different from those used by the research community. Table II shows dataset statistics.

AI-Writer. This dataset was collected from the text-generation service, AI-Writer [43]. Given a title, AI-Writer claims to generate factually accurate articles capturing recent information on the topic (based on the title). In our email communication with the service, they claim to employ custom Transformer-based LMs that are not available off-the-shelf. Since AI-Writer requires titles to generate articles, we first collect real news articles, and use the title from the real articles for generation. We evenly and randomly scraped 1000 real news articles from 20 popular news websites sampled from the RealNews [8] dataset. The list of websites is shown in Table XI in the Appendix. We verified that this dataset has no overlap with the training datasets of the defenses (Table I), based on the article publication dates. These articles form the real class of the dataset. Next, we used the real article titles to generate 1000 synthetic articles from AI-Writer. AI-Writer charges for article generation, and we spent $400 for generating 1000 articles.

ArticleForge. This dataset was collected from the Article-Forge text-generation service [44]. ArticleForge requires a set

[4] The GRUEN implementation we obtained from the authors, does not compute structure and coherence. The authors claim that this omission does not impact the metric scores significantly.

of keywords to generate an article. As per our communication with the service, they claim to use fine-tuned versions of GPT-2 [1], BERT [30] and T5 [45] to generate synthetic text. We follow a similar methodology as used for AI-Writer, and collect 1000 synthetic, and 1000 real news articles. ArticleForge charged us $57, for which they allow unlimited article generation for a month.

Kafkai. This dataset was collected from the Kafkai text-generation service [46]. Given one of 25 categories and an initial priming text, Kafkai generates a unique synthetic article that belongs to that category and is contextualized by the priming text. As per our communications with their service, Kafkai uses models from OpenAI, including GPT-2, and fine-tunes them on millions of articles to generate high quality synthetic text. We follow a similar methodology as AI-Writer and ArticleForge, and obtain context for the synthetic article generation from 1000 real articles—100 articles from 10 of the 25 available categories, *e.g.,* Cybersecurity, SEO, and Marketing. We use the first 50-100 words from each of the 1000 real articles as priming text to generate 1000 articles from Kafkai. Priming text is not included within the final article. Kafkai charged us $129 for generating 1000 articles.

RedditBot. This dataset was collected from Reddit.com. A GPT-3 powered bot posted comments under the username /u/thegentlemetre and interacted with users on /r/AskReddit, a popular subreddit on Reddit.com. Real Reddit users were initially unaware that it is a bot, and resulted in interactions with users for a week [47]. We collected 1,204 comments posted by the bot between 27th Sep, 2020 and 15th Apr, 2021, and retained 887 comments with a length greater than 192 tokens (we discard synthetic comments that are too short in interactions). We then scraped 112,296 real comments in every forum thread that contained a bot comment. To create a balanced dataset, we use a random sample of 887 real comments (with at least 192 tokens).

Linguistic quality of synthetic text in the wild We use the GRUEN metric to evaluate linguistic quality. We observe that synthetic data in the wild is comparable to synthetic text produced by the research community. More details of the text quality comparison are presented in the Appendix VII-C.

IV. Defense Performance in the Wild

A. Detection Performance

We test the 6 defenses (Section III-A) on our 4 *In-the-wild* datasets (Section III-C). Since GROVER, FAST and RoBERTa-Defense are trained for the news domain, we only test them on news domain datasets, which includes AI-Writer and ArticleForge. The remaining defenses, GLTR-BERT, GLTR-GPT2, and BERT-Defense are trained on a diverse corpus and therefore can be tested on all the datasets. We report F1 and ΔF1 (Section III-B). To compute ΔF1, we use the performance of each defense on their original test set as the baseline performance (see Table I).

Detection performance in the wild is presented in Table III. Detailed results including the Precision and Recall scores are in the Appendix (Table XII). Before we discuss the results, note that all 6 defenses achieve high detection performance (79.6% to 98.5% F1) on their original test datasets (Section III-A). Our key findings are as follows:

Finding 1: *Open-domain defenses show significant degradation in performance when applied to synthetic text in the wild, while defenses trained on data from a specific domain are able to detect In-the-wild data from that domain.* All three open-domain defenses — BERT-Defense, GLTR-BERT, GLTR-GPT2 show significant performance degradation ranging from 18.6% to 99.0% degradation in F1 score. All 3 of these defenses exhibit performance worse than a random predictor (50% F1) for at least one *In-the-wild* dataset. BERT-Defense and GLTR-GPT2 show significant degradation on all the datasets. All three news-based defenses — GROVER, RoBERTa-Defense, FAST perform well above the open-domain defenses on the news-based datasets. FAST and RoBERTa-Defense perform better on these datasets than on their original test datasets.

We further investigate the performance differences between the open-domain and the news domain defenses. Our hypothesis is that this can be attributed to distribution shift or distributional differences between data in the wild and the original datasets used to train/evaluate the defenses. To study this, we choose BERT-Defense from the open-domain category, and GROVER from the news category. We use a simple metric, average-linkage [48], to measure the distribution distance between two synthetic datasets. Given two synthetic datasets, X, and Y, we define the distribution distance as $D(X, Y)$. To represent a dataset's distribution, we randomly select 1,000 articles (or all available samples if there are fewer samples) from the dataset, and then extract the hidden state of the special [CLS] token (used as the input to the classifier) in each article from the detector as its representation. Thus X and Y each includes 1000 embedding vectors. Larger values of $D(X, Y)$ indicate a larger distribution shift.

For each defense, we compute 2 types of distance measures: (1) Distance $D(X_{train}, Y_{test})$ between its training dataset X_{train} and its original test set Y_{test} (*i.e.,* test set used in Table I). (2) Distance $D(X_{train}, Y_{wild})$ between its training dataset and each of the In-the-wild datasets Y_{wild}. We expect that $D(X_{train}, Y_{wild})$ is greater than $D(X_{train}, Y_{test})$ if the defense's performances on *In-the-wild* datasets degrade, and $D(X_{train}, Y_{wild})$ is closer to $D(X_{train}, Y_{test})$ if the detection performances are similar. Results are in Figure 1. All the $D(X_{train}, Y_{wild})$ are significantly greater than $D(X_{train}, Y_{test})$ for BERT-Defense, but not for GROVER. This observation is in line with our hypothesis.

Finding 2: *Robustly pre-trained bidirectional models,* i.e., *RoBERTa, generalize better than unidirectional models.* RoBERTa-Defense and FAST show performance improvements (F1) when applied to synthetic news samples in the wild. Both approaches use features extracted using RoBERTa, which improves over the BERT bidirectional model. The authors of RoBERTa show that BERT is significantly undertrained, and

Datasets	BERT-Defense		GLTR-GPT2		GLTR-BERT		GROVER		FAST		RoBERTa-Defense	
	F1	ΔF1	F1	ΔF1	F1	ΔF1	F1	ΔF1	F1	ΔF1	F1	ΔF1
AI-Writer	28.8	-67.6	1.6	-98.4	64.8	-18.6	87.6	+0.6	94.9	+9.1	92.1	+6.7
ArticleForge	19.7	-77.8	44.1	-55.2	85.6	+7.5	76.9	-11.7	88.5	+1.7	87.4	+1.3
Kafkai	65.9	-25.8	1.0	-99.0	41.4	-48.0	–	–	–	–	–	–
RedditBot	14.1	-84.1	61.5	-37.6	83.4	+4.8	–	–	–	–	–	–

TABLE III: Performance of the defenses, *i.e.,* F1 (%) and ΔF1 (%) of the synthetic class, on the *In-the-wild* datasets. The percentage change in F1 (ΔF1) is computed from the baseline performance of each defense. '+' means performance improvement and '-' means performance degradation. "–" (the longer minus mark) indicates experiments we ignored: We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, *i.e.,* Kafkai and RedditBot. This is because these defenses are only trained for the news domain.

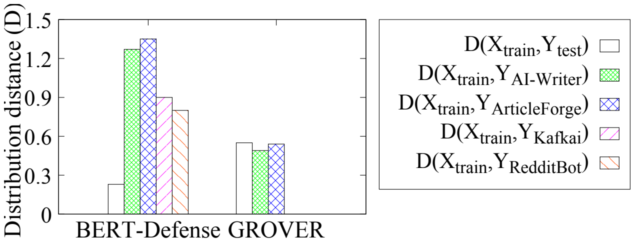


Fig. 1: Distribution distance between the training set and the test sets (including baseline test set and *In-the-wild* test sets) of BERT-Defense and GROVER.

propose changes to improve BERT's pre-training [31]. It is worth noting that GROVER, a unidirectional model, claims to perform better than bidirectional models (*e.g.,* BERT). However, our finding suggests that this claim is not true if the bidirectional model is robustly pre-trained, as in the case of RoBERTa. Moreover, a unidirectional model like GROVER is over 10x larger than the RoBERTa-base we use, in terms of number of parameters, yet it under-performs.

Another surprising result is that of GLTR-BERT performing well on ArticleForge and RedditBot (a GPT-3 dataset), but GLTR-GPT2 performs poorly on these two datasets. This means that the back-end LM model (BERT vs GPT-2) used by GLTR can have a huge impact on generalization performance.

B. Improving Performance in the Wild

Can we adapt the defenses that currently perform poorly (in the wild) to perform better on a target dataset? We investigate domain adaptation via transfer learning, *i.e.,* by fine-tuning the classifier on data from the target distribution. Language model fine-tuning has shown tremendous success for domain adaptation [49]. We consider a realistic and challenging setting, where the samples from the target distribution (for fine-tuning) are limited—in our case, as little as 10, 50 or 100 samples each for the synthetic and real class. This fits a scenario where the Internet community, including text-generation-as-a-service platforms, rapidly updates their generative models, or produces many model variants over time. In such a setting, it is hard to obtain abundant ground-truth data for attack class (synthetic samples). In such a setting, can the defender keep up?

We fine-tune the models by extending the training on the binary classification task. We consider the defenses, BERT-Defense and GROVER since they exhibit a degradation in performance (Table III).[5] While fine-tuning BERT-Defense on our small datasets, we encountered a known issue of training instability [30]. To overcome this, we employ the revitalization strategy proposed by Zhang et al. [50]. The training hyper-parameters used in BERT-Defense and GROVER fine-tuning experiments can be found in the Appendix VII-F. Results are in Table IV. Our findings are as follows:

Finding 3: *Fine-tuning with as limited as 10 In-the-wild data samples can help defenses adapt to new domains, and more samples lead to better fine-tuning performance.* Both defenses benefit from observing a few samples from both classes of the target dataset. Moreover, detection performance only improves with more fine-tuning samples.

V. Defense Against Adaptive Attackers

A. Attack Methods

To ensure real-world applicability, defenses should be effective against adaptive adversaries who are aware of the defense scheme, and can adapt the synthetic text to bypass detection. We focus on low-cost and practical adaptive attacks. Our attacks do not require a computationally expensive re-training of the attacker's generative model or creation of a surrogate/shadow defense model to craft adversarial samples. We assume a black-box setting requiring no queries to the defense scheme to craft adversarial samples. We also consider maintaining the linguistic quality of the synthetic text as the attacker's constraint. If linguistic quality is degraded significantly, it impacts the attacker's goals of misleading users, *e.g.,* synthetic fake news articles with poor linguistic quality could raise suspicion from users. These assumptions provide a more realistic setting for attackers. Our attack methods are split into two categories: (1) Attacks that change the text generation process without re-training the text generator, and (2) attacks

[5] GLTR also exhibits a degradation in performance, but we do not consider it. GLTR is not a DNN-based classifier, and is therefore not suitable for fine-tuning.

Datasets	Detection Performance (F1)							
	BERT-Defense				GROVER			
	Before Fine-tuning	# Samples for Fine-tuning			Before Fine-tuning	# Samples for Fine-tuning		
		10	50	100		10	50	100
AI-Writer	28.8	68.9	85.8	90.4	87.6	90.7	93.8	94.8
ArticleForge	19.7	85.3	90.8	93.7	76.9	86.5	95.9	97.1
Kafkai	65.9	71.0	83.8	85.0	–	–	–	–
RedditBot	14.1	71.5	90.6	95.2	–	–	–	–

TABLE IV: Detection performance in F1 (%) of BERT-Defense and GROVER on *In-the-wild* datasets before and after fine-tuning on a limited set of articles from the real and synthetic classes. "–" represents experiments we ignored: GROVER is a news domain defense, and not applicable to non-news domain datasets, *i.e.,* Kafkai and RedditBot.

that add adversarial perturbations to existing synthetic text samples to evade detection.

1) Evasion by Changing the Text Generation Process: Existing defenses are trained on synthetic text, created based on a specific decoding strategy and priming process. Our goal is to understand the robustness of defenses against changes to the text distribution triggered by varying the text generation process. Given a generative model, our idea is to craft different distributions of synthetic text samples by: (1) varying the text decoding method (and its parameters), and (2) by varying the number of priming tokens used by the model. We evaluate the linguistic quality of the adapted synthetic text using the GRUEN metric (Section III-B). *An attack is considered successful only if it degrades defense performance, while preserving linguistic quality or with limited degradation in linguistic quality.*

Each defense was trained to detect text produced using a certain decoding and priming strategy. We consider this as the baseline settings, shown in the second column of Tables V and VI. We vary the decoding strategy by considering the following methods: Top-k, Top-p (or nucleus sampling), and Temperature decoding. For each decoding strategy, we also consider different parameter settings. For Top-p, we consider several values of p in the range $[0.8, 1]$ in small increments. For Top-k, we vary k using the following values $[40, 80, 120, 160]$, and Temperature in the range $[0.7, 0.9]$. These values were determined based on standard value ranges used in prior work [8], [14], and values beyond this range resulted in significant degradation in linguistic quality.

All the defenses, except BERT-Defense is trained on unconditional text (*i.e.,* number of priming tokens is 0). BERT-Defense uses a single priming token. Priming language models with some tokens, as opposed to unconditional generation, can change the statistical and qualitative properties of generated text as the model might never generate the priming sequence on its own, *e.g.,* due to its low probability. For this attack, we generate text using a varying number of priming tokens n, where $n \in [1, 4, 8, 12]$. Each synthetic article is generated using the first n tokens from a real article. We limit the maximum number of priming tokens to 12 to minimize the amount of real text in synthetic articles.

2) Evasion by Adversarial Perturbations: We craft adversarial inputs in a black-box setting by leveraging insights unique to the synthetic text detection problem.

Crafting adversarial inputs using DFTFooler. Given a synthetic sample, our approach called DFTFooler, aims to misclassify it as real by adding adversarial perturbations to it. Unlike existing work on adversarial inputs in the text domain [15], DFTFooler requires no queries to the victim model, or a surrogate/shadow classifier to craft the perturbations. DFTFooler only requires a pre-trained LM, and several versions are publicly available today [51].

Given a synthetic article, we identify a (limited) set of words that are important for classification, and replace them with words that alter the model's prediction while preserving semantic similarity and linguistic quality. *The challenge is in identifying the important words and finding suitable replacements that alters the prediction*—we draw insights from the GLTR approach. Our insight is that generative models tend to generate the next token from the head of the distribution, thus capturing only a limited subset of the true distribution of natural language [12]. In other words, if we pass a synthetic article and a real article through a pre-trained LM (*e.g.,* BERT), the synthetic article is likely to contain many tokens which have a high probability of being generated by that language model. On the other hand, real articles will contain many low probability tokens since humans exhibit greater variation in their choice of words and this is hard for LMs to emulate. Our hypothesis is that defense schemes learn this difference between real and synthetic articles for discrimination. Therefore, to misclassify a synthetic sample, we replace a subset of the most confidently predicted words (using a pre-trained LM) by its synonyms that have lower confidence (according to the same LM).

First, DFTFooler scans a given article to choose the top N, most confidently predicted words according to a chosen LM, for replacement. The importance of a word is determined by the absolute rank of its token probability predicted by the

Defenses	Baseline Decoding Setting	Attack: Changing the Decoding Strategy								
		Top-p Decoding			Top-k Decoding			Temperature Decoding		
		Top-p	ΔR	ΔGRN	Top-k	ΔR	ΔGRN	Temp	ΔR	ΔGRN
BERT-Defense	Top-p 0.96	0.8	-13.3	+0.5	40	-12.5	+4.0	–	–	–
GLTR-GPT2	Top-k 40 Temp 0.7	0.98	-97.6	+2.0	160	-56.4	+8.8	0.9	-90.9	+4.0
GLTR-BERT	Top-k 40 Temp 0.7	0.98	-90.0	+2.0	160	-50.7	+8.8	0.9	-57.3	+4.0
GROVER	Top-p 0.94	1.0	-35.6	-4.1	160	-6.8	-0.5	–	–	–
FAST	Top-p 0.96	1.0	-9.7	-3.6	80	-2.9	+0.0	–	–	–
RoBERTa-Defense	Top-p 0.96	1.0	-22.0	-3.6	160	-2.4	+0.0	–	–	–

TABLE V: Performance of attacks that change the decoding strategy of the LM. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric (ΔR, ΔGRN) from the baseline performance of each defense on the most effective attack of each decoding strategy. '+' means performance improvement and '-' means performance degradation. "–" (the longer minus mark) indicates experiments we ignored: We only consider Temperature decoding for GLTR defenses, because the original GLTR work was evaluated using temperature-based decoding.

LM. [6] DFTFooler does not perturb stop words. More details about using a LM to find important words are described in the Appendix VII-G.

The second step is to find replacement words, while preserving semantics. To find a synonym for replacement, we build on the methodology used by TextFooler [32], and adapt it to our setting to not require queries to the victim model. For a targeted word, there are 4 sub-steps to find a valid synonym: *(1) Synonym Extraction:* We first extract a candidate set of synonyms for the targeted word as its possible word replacements. Synonym candidates are chosen according to the cosine similarity between the word embeddings of the targeted word and every other word in the vocabulary. We use word embeddings from [52] which are specially curated for finding synonyms. *(2) POS Checking*: The goal of POS checking is to assure that the grammar of the perturbed text remains the same. We will only keep synonyms with the same part-of-speech (POS) tag as the targeted word. *(3) Semantic Similarity Checking:* This step ensures that the sentence semantics before and after replacing the targeted word remain similar. Similar to TextFooler, we use the Universal Sentence Encoder (USE) [53] to compute sentence similarity. At this step, we only keep synonyms that can maintain sentence similarity scores above 0.7. *(4) Choose a synonym with low confidence as measured by a LM:* At the last step, we choose a replacement word from valid synonyms that is predicted by the LM with a low probability of ≤ 0.01. We choose the synonym with the lowest probability if multiple synonyms meet the threshold requirement. It is possible that less than N words are replaced if the semantic similarity conditions are not met. Empirically, we find that $N = 10$ works well in practice, offering a trade-off between evasion rate, and preserving linguistic quality/semantics. We implement DFTFooler using 2 back-end LMs, namely BERT and GPT2-XL. That said, the backend model is replaceable when more advanced LMs emerge in the future.

Perturbation attack baselines. We compare attack performance of DFTFooler with the following two approaches: (1) *TextFooler [32]*: TextFooler is a black-box attack that requires a large number of queries to the defense model to craft adversarial perturbations. TextFooler is highly effective against Transformer-based classifiers, and also preserves the utility of the attack by preserving the semantic content. TextFooler finds important words to replace by querying the defense model, and also queries the model to find the best replacement words. (2) *random perturbations:* This is a simple baseline that replaces random words in the article by synonyms that preserve the semantic content. This approach requires no LMs or queries to the defense model. Similar to DFTFooler, we only replace N words in an article. Such a baseline would serve to understand the benefit of replacing words based on their importance (as in our DFTFooler). DFTFooler should perform better than this baseline to be considered a useful attack.

B. Attack Evaluation

In this section, we evaluate the adaptive attacks.

Evasion by adapting the decoding process Table V presents the results for adapting the decoding strategies. Attack success is measured using the percentage change in Recall for the synthetic class, ΔR. To compute ΔR, the baseline setting is the performance on the original test dataset of each defense (Table I). A higher ΔR indicates better attack success. We only report ΔR because the real set is the same as in the baseline experiment setting (from Table I). In Table V, for each defense, we show results for the most effective attack configuration (*i.e.,* decoding method and its parameters). The most effective attack is the one with the largest degradation in ΔR, with only a minor (up to 5%) or no degradation in linguistic quality measured by the GRUEN score. We report percentage change in average GRUEN score (ΔGRN) of articles before and after

[6] For words that are split into multiple sub-tokens, we use the probability prediction of the first sub-token in the word.

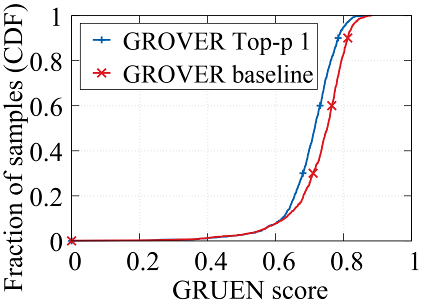


Fig. 2: The CDF of GRUEN score on GROVER's baseline test set (Top-p 0.94 used in Table I) and a dataset generated from GROVER with a decoding strategy of Top-p 1.

changing the decoding strategy. For Temperature decoding, we only show results for the GLTR defenses, because GLTR was originally evaluated using Temperature decoding. Temperature decoding is known to produce repetitive text [22], and is omitted for the other defenses. Findings are as follows:

Finding 4: *Changing the decoding strategy is a simple and effective way to break many defenses.* All defenses, except FAST, show significant degradation in ΔR under at least one of the attack strategies, ranging from 13.3% to 97.6% degradation. FAST does exhibit degradation, but to a lesser extent, compared to the other defenses, *i.e.,* degradation in ΔR ranging from 2.9% to 9.7%. This suggests that defenses like FAST are able to learn more robust features from the text. For GROVER, FAST and RoBERTa-Defense, the most effective attack (based on ΔR) happens at a Top-p value of 1.0, which is basically sampling from an untruncated distribution. However, note that the degradation in average GRUEN score is small (<5%) in these cases. Figure 2 shows the CDF of the GRUEN scores for text applied to GROVER at its baseline setting (Top-p 0.94), and when Top-p is 1.0. The two distributions are adjacent, indicating that using an untruncated distribution is not significantly degrading linguistic quality, thus providing more room for the attacker to fool defenses.

Finding 5: *Classifiers that rely solely on detecting differences in token likelihoods provided by LMs can be easily fooled by changing the decoding strategy.* We observe that GLTR-BERT and GLTR-GPT2 break down under text generated from different decoding strategies/parameters. When they encounter text generated using nucleus sampling with a Top-p value of 0.98, the recall of GLTR-BERT and GLTR-GPT2 degrades by 90.0% and 97.6%, respectively as shown in Table V. This shows that token likelihood features are highly vulnerable to attacks that change the decoding strategy. With further improvements in decoding strategies that allow more diverse text to be sampled from LMs, such defenses will only be more susceptible to these attacks.

Evasion by varying the number of priming tokens We test the defenses on text generated using a varying number of priming tokens. In Table VI, for each defense, we present results for the most effective attack configuration. Similar to the previous adaptive attack (changing decoding strategy), we use ΔR, ΔGRN to measure attack success. Our findings are as follows:

Defenses	**Baseline**	**Attack: Priming the LM**		
	#Tokens	**#Tokens**	**ΔR**	**ΔGRN**
BERT-Defense	1	0	-47.7	+6.7
GLTR-GPT2	0	4	-5.5	-4.9
GLTR-BERT	0	4	-14.8	-5.0
GROVER	0	8	-1.6	+0.3
FAST	0	12	-1.8	+0.1
RoBERTa-Defense	0	12	-1.4	+0.3

TABLE VI: Performance of attacks which prime the LM with a different number of priming tokens. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric (ΔR, ΔGRN) from the baseline performance of each defense on the most effective length of priming tokens. '+' means performance improvement and '-' means performance degradation.

Finding 6: *Defenses trained on conditionally generated text are not able to detect unconditionally generated text.* BERT-Defense which is originally trained on conditionally generated text (with a single priming token), shows over 47.7% degradation in ΔR when tested on unconditionally generated text (*i.e.,* with 0 priming tokens). All the other defenses are trained on unconditionally generated text, and show significantly less degradation, compared to BERT-Defense. We believe that this is because defenses trained on conditionally generated text learn a narrow distribution of synthetic text. We can think of LMs as being in a particular state space at each time-step. The priming tokens might lead the model into a particular state space which the model might not have reached on its own due to those 'priming tokens' being low probability tokens, and thus less likely to be sampled in an unconditional setting. Therefore, conditionally trained models might learn a different and narrower distribution of synthetic text and not generalize well to unconditionally generated text.

Evasion by adversarial perturbations. To test DFTFooler, we use a random sample of 1000 synthetic articles from the original test set of each defense, that were correctly classified. Attack success is measured using the Evasion Rate or ER metric (Section III-B). Higher ER indicates higher attack success. In addition, our attack requires preservation of semantic content. By design, our perturbation scheme achieves a USE [53] semantic similarity score ≥ 0.7, similar to TextFooler. We present our results using a small number of perturbations, *i.e.,* $N = 10$. The classifier's context window size is 512 tokens, so perturbing 10 words (or less) is a small amount of perturbation. Table VII presents the results for DFTFooler, and the baseline schemes (TextFooler and random perturbations). Our findings are as follows:

Defenses	GRN-Before	DFTFooler				Random Perturbations		TextFooler	
		BERT Backend		GPT2-XL Backend					
		ER	GRN-After	ER	GRN-After	ER	GRN-After	ER	GRN-After
BERT-Defense	0.652	1.0	0.568	1.0	0.590	0.7	0.605	50.4	0.480
GLTR-GPT2	0.686	91.3	0.594	74.2	0.633	61.6	0.642	99.7	0.670
GLTR-BERT	0.756	44.7	0.654	45.9	0.689	32.9	0.703	99.8	0.719
GROVER	0.734	59.1	0.647	43.8	0.678	30.3	0.691	99.7	0.720
FAST	0.732	24.9	0.646	23.2	0.676	14.9	0.686	99.0	0.689
RoBERTa-Defense	0.732	51.4	0.643	44.0	0.674	26.5	0.687	99.2	0.711

TABLE VII: Attack performance of the three adversarial perturbation methods (*i.e.,* DFTFooler, random perturbations, TextFooler) against the defenses, based on 10 word perturbations. GRN-Before: the average GRUEN of original datasets; GRN-After: the average GRUEN of datasets produced by adversarial perturbation attacks; ER: Evasion Rate (%).

Finding 7: *DFTFooler can successfully generate adversarial samples without requiring any information about the defense.* From Table VII, we see that DFTFooler achieves significant evasion rates ranging from 23.2% to 91.3% for all the defenses, except BERT-Defense. DFTFooler with BERT and GPT2-XL as the backend also outperforms the random perturbation attack setting for all defenses. While TextFooler outperforms DFTFooler for all defenses, it is important to note that TextFooler makes a large number of queries to the defense model to craft more effective samples whereas DFTFooler does not require queries to the model. Also, the average GRUEN scores of samples with random perturbations are comparable to the GRUEN scores of DFTFooler with GPT2-XL backend—across all defenses, the average absolute difference between GRUEN scores is only 0.014.

Finding 8: *DFTFooler using a bidirectional LM as backend, provides more effective adversarial samples.* In Table VII, DFTFooler has higher ER when it uses BERT as the backend model compared to GPT2-XL, against 4 out of 6 defenses. More specifically, DFTFooler with BERT backend shows percentage increase in ER ranging from 35.8% to 95.0% compared to random perturbations. In other words, using bidirectional context to compute token probabilities, provides better estimation of important words to be replaced. That said, we observe a slightly higher hit on GRUEN score for DFTFooler with the BERT backend compared to when GPT2-XL is used as the backend. We suspect that this is because the GRUEN score is computed, in part, by a pre-trained BERT model [42]. There is likely an overlap between the words DFTFooler chooses to perturb and the words the GRUEN metric assigns more importance to for computing sub-scores that utilize a pre-trained BERT. Figure 6 (Appendix) shows the GRUEN score distribution (CDF) for the different attacks.

Finding 9: *Increasing the number of word perturbations will improve the Evasion Rate but degrade text quality.* The attack performance of DFTFooler and random perturbations in Table VII are obtained with 10 word perturbations. To understand the impact of the number of perturbations, we experiment with a different number of word perturbations for BERT-Defense and FAST and present the results in Figure 7 in Appendix. We observe a clear trend that for both DFTFooler and random perturbations, increasing the number of perturbations will result in a higher evasion rate, but at the cost of increased degradation in the GRUEN score.

C. Towards Adversarial Robustness

Understanding robustness when using semantic features. Among all the defenses, FAST has held up more consistently against the different attacks, and also generalizes well to content in the wild. But it is still unclear what aspect of FAST contributes to its robustness. Our hypothesis is that FAST's performance can be attributed to the use of semantic features based on entities mentioned in the article. FAST models the factual structure of the article by tracking the consistency of named-entities mentioned in it. To validate this hypothesis, we analyze FAST in more detail.

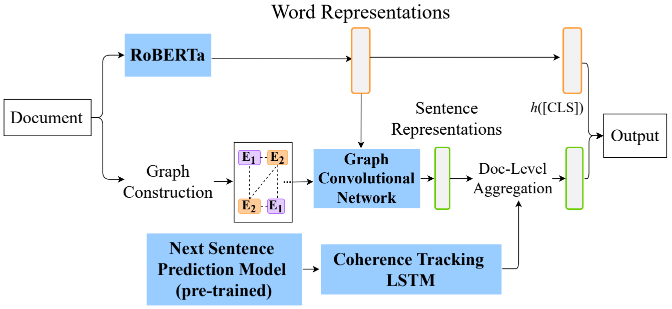


Fig. 3: An overview of the FAST pipeline.

FAST has complex internals. As shown in Figure 3, FAST comprises 4 main components: A RoBERTa-based feature extractor, a Multi-layer GCN, a Next Sentence Prediction (NSP) model and a coherence tracking LSTM. Taking a document as the input, FAST first learns contextual semantic representations for words via the RoBERTa language model. Next, a graph containing nodes representing entities in the text is created. The contextual word embeddings from RoBERTa are concatenated with Wikipedia2vec [54] entity representations to form the embedding for the entity graph. This graph

Attack Strategy	Detection Performance (Recall)		
	Distil FAST	FAST	RoBERTa-Defense
Top-p 1.0	80.1	81.6	71.5
Top-k 80	88.7	87.8	90.9
Priming Tokens 12	89.2	88.8	90.4

TABLE VIII: Detection performance (Recall) of DistilFAST, FAST and RoBERTa-Defense on attacks which change the decoding strategy used by the LM or prime it with varying numbers of priming tokens.

embedding is then fed to a multi-layer GCN to obtain graph-enhanced sentence embeddings, which are then fed to an LSTM for coherence tracking. A Next Sentence Prediction (NSP) model is used to calculate the contextual coherence score for each neighbouring sentence pair. The NSP scores are then used to compute a document-level representation from the LSTM outputs. Finally, the RoBERTa embeddings are concatenated with the document-level representation and fed to a classification layer.

To better understand FAST's superior performance, it is important to break down its complexity. We do so by running multiple ablation experiments. Models for the ablation studies are trained using the same data used to train FAST.

Ablation experiment #1: RoBERTa-Defense. We begin by considering a defense that only uses the RoBERTa language model. This is the same RoBERTa-Defense that has been evaluated in the previous sections of the paper. RoBERTa-Defense remains robust in several attacks, compared to the other defenses, but performs worse than FAST in two settings: (1) when under attack by DFTFooler (Table VII), and (2) when under attack by varying the decoding strategy (Table V). For example, RoBERTa-Defense suffers a degradation in Recall of 22.0% compared to FAST which deteriorates only by 9.7%, when it encounters text generated using Top-p 1.0 decoding. This indicates that RoBERTa is not the main source for the robustness of FAST.

Ablation experiment #2: DistilFAST. Next, we test whether the semantic features, *i.e.,* features from the entity network extracted by the GCN are the source of FAST's robustness. To do so, we create a "distilled" version of FAST, called DistilFAST, by removing the NSP task, the LSTM coherence tracker, and the Wikipedia embeddings for the GCN from FAST's pipeline. As a result, we are left with the RoBERTa model and the GCN. To create the document-level representation, we compute the element-wise sum of the sentence-level representations obtained from the GCN. To test robustness of DistilFAST, we evaluate it against : *(1)* adaptive attacks changing the generation process, *(2)* adversarial inputs based on DFTFooler and random perturbations, and (3) *In-the-wild* datasets. *If DistilFAST performs similar or better than FAST, it would suggest that use of entity-based semantic features is the key enabler for FAST's better generalization and robustness.*

Datasets	Detection Performance (F1)		
	Distil FAST	FAST	RoBERTa-Defense
AI-Writer	94.1	94.9	92.1
ArticleForge	88.2	88.5	87.4

TABLE IX: Detection performance (F1) of DistilFAST, FAST and RoBERTa-Defense on AI-Writer and ArticleForge.

Defenses	Detection Performance(Recall) with Adversarial Training			
	Adaptive Attack	Recall Before	Recall After	ΔR
BERT-Defense	Priming Token 0	44.9	85.8	+91.1
GLTR-GPT2	Top-k 160	42.8	99.2	+131.8
GROVER	Top-p 1.0	58.7	77.5	+32.0

TABLE X: Detection performance (Recall) of BERT-Defense, GLTR-GPT2, and GROVER when fine-tuned to their most effective adaptive attack setting as indicated. ΔR is percentage change of Recall from Recall before training to Recall after training.

DistilFAST against attacks changing the generation process. Table VIII shows the results. We consider attacks that change the text decoding strategy and the number of priming tokens. We test DistilFAST on the most effective attack configurations against FAST (from Tables V and VI). DistilFAST achieves a similar Recall as FAST when changing the decoding strategy to use Top-p 1.0 setting. In the other two strategies (using Top-k 80 and changing the number of priming tokens), DistilFAST even slightly outperforms FAST. We also show results for RoBERTa-Defense, which does not exhibit similar performance as FAST in one of the settings.

DistilFAST against DFTFooler and random perturbations. When the DFTFooler attack is applied to DistilFAST, we observe a 10.4% and 9.4% reduction in ER, compared to FAST, when using the BERT and GPT2 backend for DFTFooler, respectively. Similarly, against random perturbations, we observe a 13.4% reduction in ER, compared to FAST. DistilFAST is able to achieve better adversarial robustness than FAST against our adversarial perturbations.

DistilFAST against In-the-wild datasets. We evaluate DistilFAST against the *In-the-wild* datasets from the news domain, *i.e.,* ArticleForge and AI-Writer). Detection performance results are presented in Table IX. DistilFAST performs similar to FAST, suggesting that entity-based semantic features can improve generalization performance.

Our analysis leads to the following key finding:

Finding 10: Semantic features that capture the factual structure of the text, *i.e.,* entity-level features, provides robustness against adaptive attacks and better generalization performance.

Adversarial training to enable robustness against adaptive attacks. We investigate whether a defender can recover from an adaptive attack via *adversarial training*, *i.e.,* by training a defense on a set of known adversarial samples to build resilience against similar adversarial samples. We start by studying recovery from adaptive attacks that change the text generation process. We fine-tune BERT-Defense, GROVER and GLTR-GPT2 [7] on new samples generated from their most effective adaptive attack setting (Tables V and VI). We use 1,000 new articles each for both the synthetic and real class for adversarial training. We then evaluate the adversarially trained models against their original adaptive attack dataset. As shown in Table X, we observe that the fine-tuned BERT-Defense, GLTR-GPT2 and GROVER achieve 91.1%, 131.8% and 32.0% increase in ΔR, respectively. Therefore, the fine-tuned models are able to recover from the attack.

Next, we explore adversarial training to recover from our DFTFooler attack. We use RoBERTa-Defense for this experiment. From our dataset of 1,000 adversarial samples from DFTFooler, we use a random set of 500 samples for adversarial fine-tuning, and test the adversarially trained RoBERTa-Defense on the remaining 500 adversarial samples. DFTFooler becomes ineffective as its evasion rate drops from 51.4% on RoBERTa-Defense (in Table VII) to 1.6% on the adversarially fine-tuned RoBERTa-Defense.

Fine-tuning a defense towards a specific attack is effective, but the defender may have to frequently adapt their defenses against newer adaptive strategies or their variants. On the other hand, by using robust semantic features, one can potentially build in resilience in an attack-agnostic way.

VI. Discussion

Deepfake text detection vs. deepfake image detection. Deepfake image detection and deepfake text detection both aim to detect synthetic content generated by deep generative models. That said, these two fields hardly share the same set of technical methodologies in generating and detecting synthetic content due to the discrete nature of text. Similar to the text domain, researchers have also demonstrated impressive performance in detecting deepfake images [55], [56]. This has prompted work investigating the adversarial robustness of deepfake image detectors [57]–[59]. In a similar way, our work is the first to systematically explore real-world performance and adversarial robustness of deepfake text detectors. While the image domain has primarily focused on adversarial perturbations, our work also explores low-cost evasion strategies that change the content generation process. Note that the adversarial attacks studied for the image domain are not directly applicable to the text domain. If we compare detection schemes in the two domains, more progress has been made on understanding the use of semantic features for deepfake image detection [60]–[63]. An example is a method that looks for eye blinking artifacts [64] to detect deepfake images. However, in the text domain, FAST is the only approach that leverages semantic features (entity-based), and there is scope for more work to be done in this direction.

Further exploring semantic layer methods for synthetic text detection. Producing semantically consistent text is still a challenging task for language models [65]. Therefore, we expect to find differences in the semantic information embedded in synthetic text when compared to real text. In this study, we show that FAST leverages such imperfections in the semantic structure of the text to differentiate between synthetic and real text. Using semantic information will also raise the cost of producing synthetic text that can bypass such defenses. In practice, the attacker may want to preserve the semantic content, *e.g.,* spreading vaccine disinformation, while creating evasive samples, thus making it harder to evade detection. One direction for future work is to leverage *knowledge graphs* to extract richer semantic features.

Ethics. Our work involved collecting synthetic text data from *text-generation-as-a-service* platforms and from Internet forums. We spent $586 to collect the articles from the services. All the services we study claim that the synthetic articles can be used for white hat SEO. Regardless of the legal status of these services, the benefits gained from understanding how well the state-of-the-art defenses perform on synthetic text generated from them outweighs the potential harms arising from injecting money into these services. Our work proposes as well as evaluates existing defenses against adversarial inputs. This was done in a controlled lab setting, and no deployed models were attacked in this process.

VII. Conclusion

To the best of our knowledge, this work presents the first systematic evaluation of deepfake text defenses to assess their real-world applicability. We evaluated state-of-the-art synthetic text defenses on real-world datasets and against adaptive attackers. We find that open-domain detection schemes fail to generalize to *In-the-wild* synthetic text, and that most defenses are not robust under adversarial settings. We also presented DFTFooler, a novel adversarial sample crafting scheme, that can degrade the performance of existing defenses without requiring any queries to the victim model nor a surrogate classifier. Our detailed analysis of the most robust defense (FAST) indicates that utilizing semantic information in the text samples can lead to better robustness and generalization performance in the wild.

Acknowledgment

This research was supported by startup funds from Virginia Tech. Zain Sarwar was funded by the Higher Education Commission (HEC) Pakistan National Center for Cyber Security (NCCS) grant at LUMS. We thank Neal Mangaokar for discussions of text decoding strategies and their impact on text generation.

[7] Since GLTR-GPT2 is not DNN-based, we instead train a new GLTR-GPT2 model from scratch with samples generated under the attack setting.

References

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language Models are Unsupervised Multitask Learners," *OpenAI blog*, 2019.

[2] T. Brown et al., "Language Models are Few-Shot Learners," in *Proc. of NeurIPS*, 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. of NeurIPS*, 2017.

[4] P. Xu, M. Patwary, M. Shoeybi, R. Puri, P. Fung, A. Anandkumar, and B. Catanzaro, "MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models," in *Proc. of EMNLP*, 2020.

[5] F. Mi, L. Chen, M. Zhao, M. Huang, and B. Faltings, "Continual Learning for Natural Language Generation in Task-oriented Dialog Systems," in *Proc. of EMNLP*, 2020.

[6] H. Zhang, J. Xu, and J. Wang, "Pretraining-Based Natural Language Generation for Text Summarization," in *Proc. of CoRR abs/1902.09243*, 2019.

[7] L. Leppänen, M. Munezero, M. Granroth-Wilding, and H. Toivonen, "Data-Driven News Generation for Automated Journalism," in *Proc. of INLG*, 2017.

[8] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending Against Neural Fake News," in *Proc. of NeurIPS*, 2019.

[9] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, and B. Y. Zhao, "Automated Crowdturfing Attacks and Defenses in Online Review Systems," in *Proc. of CCS*, 2017.

[10] A. Das and R. Verma, "Automated email Generation for Targeted Attacks using Natural Language," in *Proc. of TA-COS*, 2018.

[11] K. McGuffie and A. Newhouse, "The Radicalization Risks of GPT-3 and Advanced Neural Language Models," in *Proc. of CoRR abs/2009.06807*, 2020.

[12] S. Gehrmann, H. Strobelt, and A. M. Rush, "GLTR: Statistical Detection and Visualization of Generated Text," in *Proc. of ACL*, 2019.

[13] D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck, "Automatic Detection of Generated Text is Easiest when Humans are Fooled," in *Proc. of ACL*, 2020.

[14] W. Zhong, D. Tang, Z. Xu, R. Wang, N. Duan, M. Zhou, J. Wang, and J. Yin, "Neural Deepfake Detection with Factual Structure of Text," in *Proc. of EMNLP*, 2020.

[15] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP," in *Proc. of CoRR abs/2005.05909*, 2020.

[16] S. Chen, N. Carlini, and D. Wagner, "Stateful Detection of Black-Box Adversarial Attacks," in *Proc. of ACM AISec*, 2020.

[17] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Defending Black-Box Adversarial Attacks on Deep Neural Networks," in *Proc. of CoRR abs/2006.14042*, 2020.

[18] J. Byun, H. Go, and C. Kim, "On the Effectiveness of Small Input Noise for Defending Against Query-based Black-Box Attacks," in *Proc. of WACV*, 2022.

[19] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proc. of ACL*, 2016.

[20] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical Neural Story Generation," in *Proc. of ACL*, 2018.

[21] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The Curious Case of Neural Text Degeneration," in *Proc. of CoRR abs/1904.09751*, 2019.

[22] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[23] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in Data-to-Document Generation," in *Proc. of EMNLP*, 2017.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Neurocomputing*, 1987.

[25] A. Graves, "Long Short-Term Memory," *Springer*, pp. 37–45, 2012.

[26] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *Int. J. Uncertain. Fuzziness Knowledge-Based Syst.*, vol. 6, no. 02, pp. 107–116, 1998.

[27] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE trans. neural netw.*, vol. 5, no. 2, pp. 157–166, 1994.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *Proc. of ICLR*, 2015.

[29] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. of NAACL*, 2019.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," in *Proc. of CoRR abs/1907.11692*, 2019.

[32] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment," in *Proc. of AAAI*, 2020.

[33] L. Yuan, X. Zheng, Y. Zhou, C.-J. Hsieh, and K.-W. Chang, "On the Transferability of Adversarial Attacks against Neural Text Classifier," in *Proc. of EMNLP*, 2021.

[34] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "BERT-ATTACK: Adversarial Attack Against BERT Using BERT," in *Proc. of EMNLP*, 2020.

[35] "Common crawl: A nonprofit 501 organization that crawls the web and freely provides its archives and datasets to the public." https://commoncrawl.org/, 2019.

[36] "GROVER: Code for defending against neural fake news," https://github.com/rowanz/grover, 2019.

[37] "GLTR: Giant Language Model Test Room," https://github.com/HendrikStrobelt/detecting-fake-text, 2019.

[38] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.

[39] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *Proc. of EMNLP*, 2018.

[40] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proc. of EMNLP*, 2016.

[41] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "RACE: Large-scale ReAding Comprehension Dataset From Examinations," in *Proc. of EMNLP*, 2017.

[42] W. Zhu and S. Bhat, "GRUEN for Evaluating Linguistic Quality of Generated Text," in *Proc. of EMNLP*, 2020.

[43] "AI-Writer text generation service," http://ai-writer.com/, 2020.

[44] "Article Forge: Get HIGH QUALITY Content In 60 Seconds," https://www.articleforge.com/, 2020.

[45] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *JMLR*, vol. 21, pp. 1–67, 2020.

[46] "Kafkai: AI Writer & AI Content Generator," https://kafkai.com/, 2020.

[47] W. D. Heaven, "A GPT-3 bot posted comments on Reddit for a week and no one noticed (MIT Technology Review)," https://www.technologyreview.com/2020/10/08/1009845/a-gpt-3-bot-posted-comments-on-reddit-for-a-week-and-no-one-noticed/, 2020.

[48] B. Moseley and J. Wang, "Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search," in *Proc. of NeurIPS*, 2017.

[49] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," in *Proc. of ACL*, 2018.

[50] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting Few-sample BERT Fine-tuning," in *Proc. of CoRR abs/2006.05987*, 2020.

[51] "Hugging Face: The AI community building the future," https://huggingface.co/models, 2016.

[52] N. Mrkšić, D. O. Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting Word Vectors to Linguistic Constraints," in *Proc. of NAACL-HLT*, 2016.

[53] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar *et al.*, "Universal Sentence Encoder," in *Proc. of CoRR abs/1803.11175*, 2018.

[54] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto, "Wikipedia2Vec: An Efficient Toolkit for Learning and

Visualizing the Embeddings of Words and Entities from Wikipedia," in *Proc. of EMNLP*, 2020.

[55] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, "Leveraging Frequency Analysis for Deep Fake Image Recognition," in *Proc. of ICML*, 2020.

[56] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *Proc. of CVPR*, 2020.

[57] N. Carlini and H. Farid, "Evading Deepfake-Image Detectors With White- and Black-Box Attacks," in *Proc. of CVPR*, 2020.

[58] A. Gandhi and S. Jain, "Adversarial Perturbations Fool Deepfake Detectors," in *Proc. of IJCNN*, 2020.

[59] X. Cao and N. Z. Gong, "Understanding the Security of Deepfake Detection," in *Proc. of CoRR abs/2107.02045*, 2021.

[60] Y. Li, M.-C. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking," *IEEE WIFS*, 2018.

[61] Y. Li and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts," in *Proc. of CoRR abs/1811.00656*, 2018.

[62] U. A. Ciftci, I. Demir, and L. Yin, "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals," *IEEE TPAMI*, 2020.

[63] F. Matern, C. Riess, and M. Stamminger, "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations," in *Proc. of IEEE WACVW*, 2019.

[64] T. Jung, S. Kim, and K. Kim, "DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern," *IEEE Access*, vol. 8, pp. 83 144–83 154, 2020.

[65] Y. Dou, M. Forbes, R. Koncel-Kedziorski, N. A. Smith, and Y. Choi, "Is GPT-3 Text Indistinguishable from Human Text? Scarecrow: A Framework for Scrutinizing Machine Text," in *Proc. of CoRR abs/2107.01294*, 2021.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proc. of ACL*, 2002.

[67] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," in *Proc. of ACL Workshop*, 2005.

[68] G. Doddington, "Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics," in *Proc. of ICHLT*, 2002.

[69] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proc. of ACL*, 2004.

[70] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proc. of ICLR Workshop*, 2013.

[71] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proc. of EMNLP*, 2014.

[72] V. Rus and M. Lintean, "An Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics," in *Proc. of ITS*, 2012.

[73] T. Landauer and S. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge." *Psychological Review*, vol. 104, no. 2, p. 211, 1997.

[74] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau, "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses," in *Proc. of ACL*, 2017.

[75] L. Huang, Z. Ye, J. Qin, L. Lin, and X. Liang, "GRADE: Automatic Graph-Enhanced Coherence Metric for Evaluating Open-Domain Dialogue Systems," in *Proc. of EMNLP*, 2020.

[76] S. Mehri and M. Eskenazi, "USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation," in *Proc. of ACL*, 2020.

[77] A. Warstadt, A. Singh, and S. R. Bowman, "Neural Network Acceptability Judgments," *TACL*, vol. 7, pp. 625–641, 2019.

[78] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From Word Embeddings to Document Distances," in *Proc. of ICML*, 2015.

[79] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," in *Proc. of CoRR abs/1909.11942*, 2020.

[80] D. I. Adelani, H. Mai, F. Fang, H. H. Nguyen, J. Yamagishi, and I. Echizen, "Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-Based Detection," in *Proc. of AINA*, 2020.

[81] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps *et al.*, "Release Strategies and the Social Impacts of Language Models," in *Proc. of CoRR abs/1908.09203*, 2019.

[82] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi, "TweepFake: About detecting deepfake tweets," *PLoS ONE*, vol. 16, no. 5, p. e0251415, 2021.

[83] "GridSearchCV in ScikitLearn," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, 2011.

[84] "Logistic Regression Classifier in ScikitLearn," https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, 2011.

Appendix

A. Metrics for Evaluating Linguistic Quality

Besides GRUEN, we surveyed other text quality metrics and categorized them into the following groups: word-based metrics, embedding-based metrics, training-based metrics, and dialog-based metrics. Here we explain the main features of each metric category and their limitations given our usage scenario. Both word-based and embedding-based metrics require human reference text for evaluating text quality. Word-based metrics compute text quality based on the word or n-gram overlap between the evaluated text and the reference text, e.g., BLEU [66], METEOR [67], NIST [68], and ROUGE [69]. Word-based metrics rely largely on word-level matches. Such similarities can be better captured by word embeddings such as Word2Vec [70] and GloVe [71]. Thus, an alternative to matching words is to compare the similarity between the embeddings of words in the evaluated text and the reference, e.g., Greedy Matching [72] and Embedding Average metric [73]. The main limitation of these metrics is that all of them require references from the real class, which are not available in our case. Instead of comparing with human generated gold standard references, training-based metrics contain learnable components that are trained specifically for the task of automatic evaluation, e.g., ADEM [74]. However, training this model requires human annotations which were not available to us. Other text quality metrics such as GRADE [75] and USR [76] were designed for evaluating the quality of synthetic dialogs, and are difficult to transfer across usage domains. Besides automatic evaluation metrics, another way to evaluate text quality is to conduct a human study to annotate the quality of documents. However, given the large number of datasets we evaluated, this was not realistic for us to do.

B. A Detailed Description of GRUEN Score

GRUEN, proposed by Zhu *et al.* [42], is an unsupervised and reference-less text quality metric. Zhu *et al.* show that GRUEN is more correlated with human judgement of text quality than any other existing metric. The GRUEN score of an article is computed by aggregating the following sub-scores:

Grammaticality. This is computed by combining two sub-scores: Perplexity and grammar acceptance. Perplexity is computed using a BERT model whereas the grammar acceptance score is computed by fine-tuning a BERT on the CoLA dataset [77] which contains labelled examples of grammatically correct and incorrect sentences.

Non-redundancy. This metric computes whether a document contains excessively repeated sentences, phrases and instances where proper nouns were used instead of pronouns.

This is done by computing four inter-sentence syntactic features: length of the longest common substring, count of the longest common words, edit distance and the number of common words in a document.

Focus. This score looks at the semantic similarity between adjacent sentences as a measure of discourse focus. It is computed via the Word Mover Similarity [78] for adjacent sentences.

Structure and coherence. This is calculated by computing the loss on the Sentence-Order-Prediction [79] task as it models the inter-sentence coherence in a document. In the code provided by the authors of the GRUEN metric, this component was not included.

C. Linguistic Quality of Synthetic Text in the Wild

We use the GRUEN metric to evaluate linguistic quality. Figure 4 shows the CDF of GRUEN scores for *In-the-wild* synthetic samples, compared with synthetic text produced by the research community, which includes GPT2-XL and GROVER. We can see that data in the wild is comparable or better than synthetic text produced by the research community.

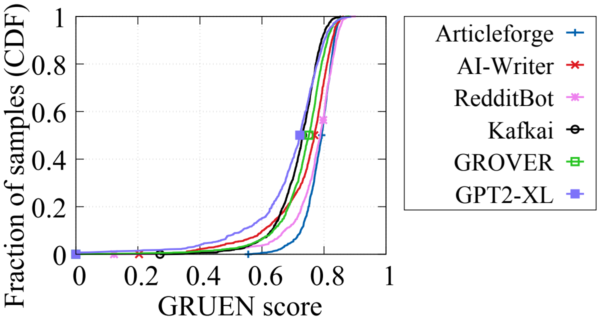


Fig. 4: The CDF of GRUEN on the 4 *In-the-wild* datasets and 2 datasets generated from GROVER and GPT2-XL.

D. Other Defenses

There are a few other defenses that are not considered in our study. Yao *et al.* [9] in 2017, proposed a method to detect LSTM-generated synthetic reviews (restaurant reviews targeting Yelp). While the proposed method works well against LSTM-generated text, synthetic text generation has advanced significantly since Transformers were introduced. Yao's method uses an LSTM-based supervised approach to detect synthetic text. We omit this defense because our preliminary evaluation of this approach on synthetic text produced by Transformers yielded unsatisfactory results. We upgraded Yao's approach to use a Transformer-based classifier (instead of an LSTM model), and trained the model on 5000 real articles from the RealNews dataset, and 5000 articles produced by GROVER. Unfortunately, Yao's approach only achieves an F1-score of 68% in detecting GROVER generated text.

Adelani *et al.* [80] present several classifiers, *e.g.,* GLTR, GROVER and an OpenAI GPT-2 based Detector [81] to detect synthetic reviews generated using GPT2-Small. Fagni *et al.* [82] develop several classifiers based on Markov Chains, RNNs, LSTMs, and GPT-2 to detect synthetic tweets on Twitter. We do not study both approaches because many of the state-of-the-art methods such as GROVER, GLTR and other Transformer-based models are already considered in our work. Moreover, based on our preliminary investigation of Yao *et al.*'s work, we found that RNN/LSTM-based models are not promising approaches to detect synthetic text produced by advanced models like Transformers.

E. Applying Grid Search for Training GLTR

To tune the hyperparameters of GLTR defenses (based on logistic regression classifier), we apply grid search to the training process of the GLTR defenses, which includes GLTR-GPT2 and GLTR-BERT. We use *GridSearchCV* [83] which is built into scikit-learn to select the model. Given a set of parameter values, GridSearchCV can exhaustively consider all parameter combinations, fit the model on the training set, and select the best model. To train GLTR defenses, we apply grid search on the following hyperparameters of the logistic regression classifier, *i.e.,* "solver", "penalty" and "C". Different choices of 'solver", "penalty" and "C" can result in differences in model performance [84]. We use GridSearchCV to loop through the following parameter values—['newton-cg', 'lbfgs', 'liblinear'], ['l2'], [100, 10, 1.0, 0.1, 0.01] for "solver", "penalty" and "C", respectively.

F. Details of Fine-tuning Experiments

BERT-Defense fine-tuning experiments with In-the-wild samples. To improve BERT-Defense's detection performance on *In-the-wild* datasets, we fine-tune BERT-Defense with a limited set of *In-the-wild* samples (*i.e.,* 10, 50 and 100 samples) in Section IV-B. We follow the general guidelines of transfer learning to fine-tune BERT-Defense [8]. We set batch size as 4, and fine-tune the BERT-Defense model for 8 epochs. While doing this experiment, we encountered a known problem of instability of fine-tuning BERT on small datasets [50]. To overcome this problem, we employ the revitalization strategy proposed by Zhang et al. [50]. We also tune certain training parameters to improve our results. Specifically, we use the "*adamw_torch*" optimizer, and set the "*weight_decay*" and "*warmup_ratio*" to 10e-5 and 0.3, respectively.

GROVER fine-tuning experiments with In-the-wild samples. To improve GROVER's detection performance on *In-the-wild* datasets, we fine-tune GROVER with a limited set of *In-the-wild* samples (10, 50 and 100 samples). We use a batch size of 4, and fine-tune the GROVER model for 3 epochs.

G. More Details of the DFTFooler Pipeline

Identifying a set of important words to be replaced via a LM. DFTFooler scans a given article to choose the top N most confidently predicted words according to the chosen LM, for replacement. For example, GPT-2 is a standard left-to-right language model, and after tokenizing the articles into

[8] https://huggingface.co/docs/transformers/training#finetune-a-pretrained-model

Tools	Websites
ArticleForge	politico.com, usatoday.com, deseretnews.com, hollywoodreporter.com, theatlantic.com, nbcphiladelphia.com, reuters.com, reuters.com, dailymail.co.uk, theguardian.com
AI-Writer	arabnews.com, bbc.com, dailymail.co.uk, dailytimes.com.pk, dawn.com, deseret.com, esquire.com, gizmodo.com, hollywoodreporter.com, mashable.com, nbcphiladelphia.com, nj.com, politico.com, reuters.com, theatlantic.com, theglobeandmail.com, thenorthernecho.co.uk, thisismoney.co.uk, usatoday.com, thedailystar.net

TABLE XI: Websites used to scrape real news articles for ArticleForge and AI-Writer datasets.

Datasets	BERT-Defense			GLTR-GPT2			GLTR-BERT			GROVER			FAST			RoBERTa-Defense		
	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
AI-Writer	28.8	73.1	17.9	1.6	100	0.8	64.8	77.5	55.7	87.6	78.4	99.3	94.9	91.0	99.2	92.1	86.9	98.1
ArticleForge	19.7	59.0	11.8	44.1	97.3	28.5	85.6	76.3	97.6	76.9	62.6	99.8	88.5	84.7	92.7	87.4	80.1	96.3
Kafkai	65.9	90.0	52.0	1.0	62.5	0.5	41.4	57.6	32.3	–	–	–	–	–	–	–	–	–
RedditBot	14.1	25.8	9.7	61.5	99	44.6	83.4	72.8	97.5	–	–	–	–	–	–	–	–	–

TABLE XII: Performance of the defenses on the *In-the-wild* datasets. We present F1 score (F1), Precision (P), and Recall (R) of the synthetic class in percentages. We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, which includes Kafkai and RedditBot. This is because these defenses are only trained for the news domain.

a sequence of tokens $\{x_1, x_2, ..., x_i\}$, GPT-2 can compute the prediction probability of token x_i, using Equation 1, *i.e.,* $p(x_i|x_0, x_1, ..., x_{i-1})$. For simplicity, assume that each token is a word in the article. At each step in the sequence, a word is assigned a rank among all the words in the vocabulary based on its prediction probability score (higher probability leads to higher rank). In practice, a word can be tokenized into multiple tokens. For words that are tokenized into multiple tokens by the tokenizer, DFTFooler uses the probability score of the word's first subtoken as the word's probability score. This way, each word in the sequence is assigned a rank based on this probability score. We eventually choose the set of top N most highly ranked words.

H. List of Real News Websites

As explained in Sec III-C, our *In-the-wild* dataset contained an equal number of real and fake articles. For generating news articles, we used two text generation services, namely AI-Writer and ArticleForge.

ArticleForge can generate fake articles with a set of provided keywords. In this case, we collected 1000 real news articles from 10 news websites, and used keywords from them to generate 1000 fake news articles.

On the other hand, AI-Writer requires a title to generate an article. Similar to the method used for ArticleForge, we scraped 1000 news articles from a list of 20 news websites, and used their titles to generate synthetic articles. In both cases, the list of news websites are listed in Table XI.

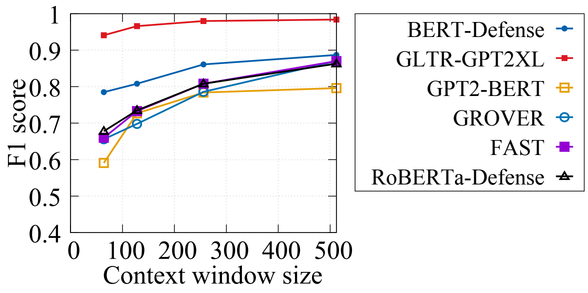


Fig. 5: Detection performances of the defenses with different context window sizes, *i.e.,* 64, 128, 256, 512 tokens.

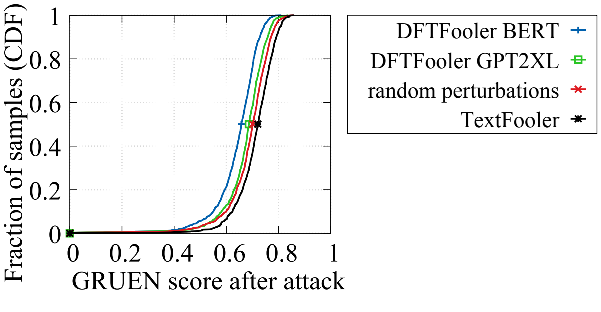
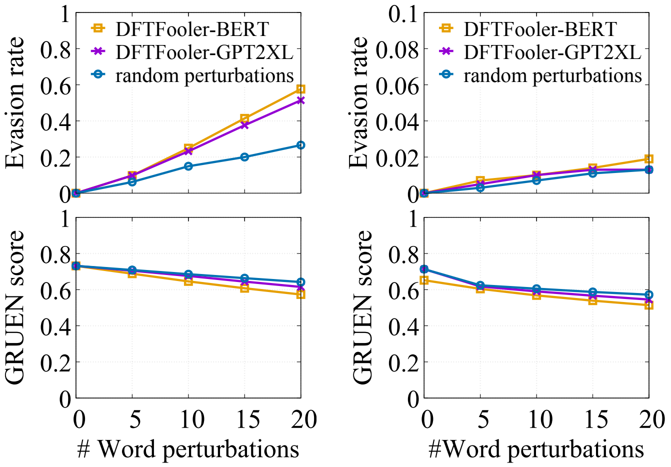


Fig. 6: The CDF of GRUEN score on perturbed text produced by different adversarial perturbation methods, *i.e.,* DFTFooler, random perturbations, and TextFooler, based on attacking RoBERTa-Defense (in Table VII).



(a) Attacking FAST.

(b) Attacking BERT-Defense.

Fig. 7: Evasion rate and average GRUEN score of perturbed text achieved by DFTFooler and random perturbations when attacking (a) FAST and (b) BERT-Defense, based on 5, 10, 15, and 20 word perturbations.