NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

GAN-GPTZERO: A GENERIC GENERATED TEXT DETECTION ARCHITECTURE

HAOJIE MU
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING
2023

GAN-GPTZERO: A GENERIC GENERATED TEXT DETECTION ARCHITECTURE

HAOJIE MU

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN
COMPUTER CONTROL & AUTOMATION

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

mhjie

03.Oct.2023

. . . .

Date

. . . .

HAOJIE MU

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

03.Oct.2023	[illegible]
....	....
Date	Ling Keck-Voon

Authorship Attribution Statement

This thesis does not contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

03.Oct.2023

. . . .

Date

mhjie

. . . .

HAOJIE MU

Table of Contents

Abstract

With the burgeoning usage of AI language models like ChatGPT across numerous sectors including corporations, educational institutions, and research entities, concerns over issues of plagiarism and propagation of false information have simultaneously escalated. To mitigate these issues, some preliminary solutions such as DetectGPT and RoBERTa detectors have been introduced. However, the accuracy of these methodologies is markedly influenced by several factors, including the length of the text, the similarities in writing patterns between ChatGPT and a particular user demographic, and the vast array of topics encompassed. Current research largely focuses on creating a universal detector, albeit with limited success due to these aforementioned challenges. This dissertation deviates from this generic approach, proposing a Generative Adversarial Network based GPT detector, dubbed as GAN-GPTZero, offering enhanced accuracy with a meticulously designed architecture. Following an in-depth literature review and the formal presentation of our research objectives, comprehensive set of experiments have been conducted to validate the effectiveness of GAN-GPTZero. This dissertation offers a nuanced understanding of the underlying mechanisms of various detectors, while both theoretically and empirically demonstrating how GAN-GPTZero outperforms existing state-of-the-art solutions.

Keywords: GPT, GAN, Transformers, Large Language Model, Detector.

Acknowledgements

Special thanks are conveyed to my research project supervisor Prof. Ling for his continual guidance, support and incisive recommendations concerning the scope of the research.

Additionally, but by no means least, my heartfelt appreciation extends to my colleagues, family, and friends, whose emotional support and understanding were essential throughout this journey. Their encouragement during challenging periods and joy in times of success provided a balance that was crucial to maintaining my perseverance in this research.

This dissertation significantly leverages the codebase and database created by Huggingface. The conceptual framework of the study is inspired by the Detect-GPT and the RoBERTa Detector.

The completion of this research signifies not only an academic achievement but also the combined efforts of every individual and institution acknowledged here. I am deeply grateful for the collective support and contributions that have made this accomplishment possible.

Acronyms

AUROC	Area Under the Receiver Operating Characteristic curve
GAN	Generative Adversarial Network
GPT	Generative Pre-trained Transformer
BERT	Bidirectional Encoder Representations from Transformers
TPR	True Positive Rate
FPR	Frue Positive Rate
NLP	Natural Language Processing
LLM	Large Language Model
MLM	Masked Language Model
RNN	Recurrent Neural Networks
LSTM	Long Short-term Memory
GRU	Gated Recurrent Units
NLL	Negative Log-Likelihood
CE-Loss	Cross-Entropy Loss
BCE-Loss	Binary Cross-Entropy Loss

Symbols

x	A token
z	A prompt, consisted of words
Z	A batch of prompts
s	A sentence, consisted of words
S	A text, consisted of a batch of sentences
n	The length of a sentence
m	The batch size
s^h	A human written sentence
s^m	A machine generated sentence
S^h	A human written text
S^m	A machine generated text
$\tilde{s}$	A perturbed sentence
$\hat{s}$	A prediction of a sentence
$\tilde{S}$	A perturbed text
$\hat{S}$	A prediction of a text
G	A Generator
P	A Perturbator
D	A Discriminator
$\mathcal{H}$	The distribution of human written text
$\mathcal{M}$	The distribution of machine generated text
$\mathbb{P}$	Predicted possibility of a Generator
$\mathcal{P}$	Predicted possibility of a Perturbator
P	Predicted possibility of a Discriminator
$\mathcal{A}$	Condensed formulation of Generator Loss
$\mathcal{B}$	Condensed formulation of Discriminator Loss of GAN-DetectGPT variant
$\mathcal{C}$	Condensed formulation of Discriminator Loss of GAN-RoBERTa variant
f	Discriminator prediction score
F	A batch of prediction scores
γ	Prediction threshold
f_γ^h	Predicted as human written
f_γ^m	Predicted as machine generated
f_{-}	Negative Log-Likelihood Loss

List of Figures

List of Tables

Chapter 1

Introduction

This introductory chapter serves as the gateway to the comprehensive examination of GAN-GPTZero. It lays down the contextual background, highlights the importance and relevance of the subject in today's global landscape. This chapter outlines the primary objectives of GAN-GPTZero and incorporates a preliminary discussion on the underlying theories in associated domains, along with its existing limitations. Furthermore, it lists the subsequent chapters and furnishes a detailed exposition of the structural framework governing this dissertation.

1.1 Authenticity of Large Language Models

Among large language models that have emerged, the Generative Pre-trained Transformer (abbreviated as GPT) [1], stands out as one of the most illustrious and widely recognized. GPT's versatility has been showcased across a myriad of natural language processing (abbreviated as NLP) tasks, from text classification and sentiment analysis to machine translation and question answering [2] [3]. This adaptability has propelled GPT to the forefront of the field, enabling new research directions and practical applications. Yet, its unparalleled capacity to generate coherent and contextually relevant text has also sparked serious ethical considerations. Concerns have arisen about the potential misuse of

GPT, particularly in crafting fake news, propaganda, and disinformation, causing a complex dialogue about responsibility, governance, and control in the deployment of such transformative technologies.

To address these concerns, researchers have proposed several methods to detect GPT-generated text, including the use of statistical methods, machine learning algorithms, and linguistic features. One popular approach is the use of zero-shot learning techniques without requiring any labeled data from that generator.

DetectGPT [4] is another zero-shot learning framework designed to detect text generated by GPTs. It is built upon the idea that GPTs exhibit unique patterns and characteristics that can be leveraged to identify their generated text. DetectGPT utilizes a self-supervised learning approach, which means it does not require any labeled data from the target domain (i.e., GPT-generated text). Instead, it relies on a pretraining task to learn representations that are useful for detecting GPT-generated text. Due to its label-free characteristic, it experiences a diminished level of representation that in turn compromises the accuracy of DetectGPT.

1.2 What are Large Language Models

The landscape of NLP has been radically transformed by the advent of large-scale transformer-based language models. One of the pioneering efforts in this space was the Transformer model introduced by [5], which laid the foundation for many subsequent developments.

In the realm of NLP, transformer-based architectures, often categorized under the umbrella term large language models (abbreviated as LLM), demand an extensive computational budget, both in terms of time and financial resources. Despite these considerable requirements, which include an exponential increase in the number of parameters, these models have demonstrated remarkable performance across a variety of tasks, thus justifying the investment.

The models introduced in the following sections are all classified under the category of LLMs and will be of significant relevance in the subsequent discussions and analyses presented in this thesis.

GPT series has been particularly influential, with models like GPT-2 [1] and GPT-3 [3] setting new standards in various language tasks. These models, built on a transformer architecture, have shown that scaling up model size can lead to substantial gains in performance [6].

At the same time, other significant contributions have emerged. Bidirectional Encoder Representations from Transformers (abbreviated as BERT) [7] introduced a bidirectional training strategy that enhanced understanding of context within text. This approach has spawned numerous variations and adaptations, such as RoBERTa [8], which is a finetuned BERT's training methodology; and ERNIE [9], which further incorporates knowledge integration.

Furthermore, models like T5 (Text-to-Text Transfer Transformer) [10] have explored the idea of treating all NLP tasks as text-to-text tasks, unifying various language processing tasks under a common framework. Meanwhile, XLNet [11] has presented an alternative to BERT's bidirectional approach, using a permutation-based training objective to capture more comprehensive information from the text.

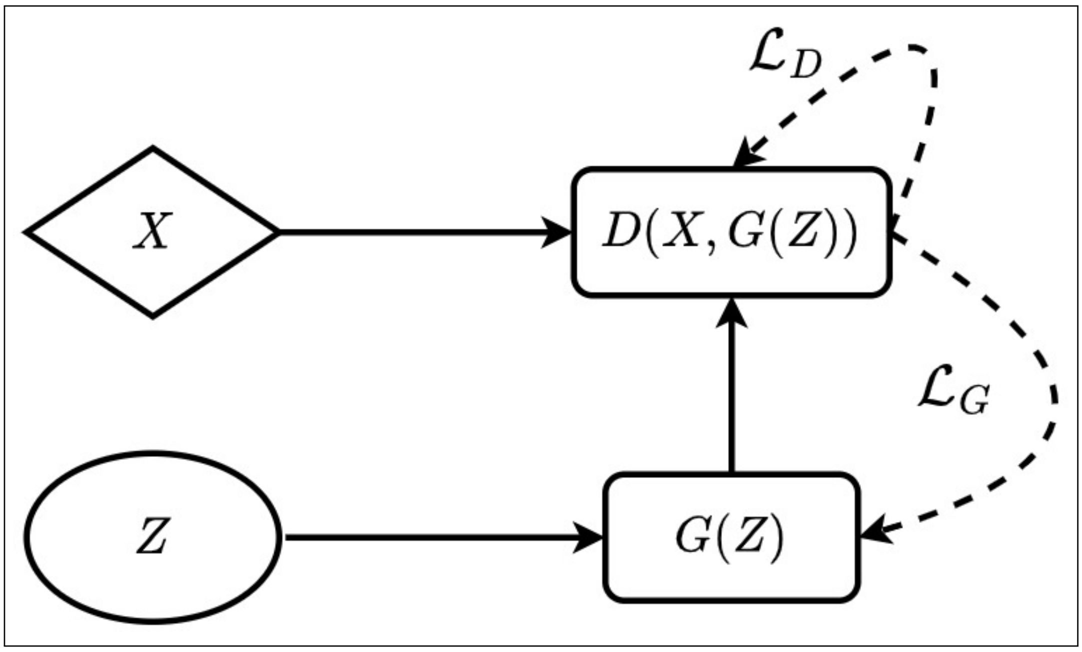


Figure 1.1: The Generative Adversarial Network.
The Generative Adversarial Network possesses the capacity to enhance both the Generator G and the Discriminator G through the implementation of Generator Loss $\mathcal{L}_G$ and Discriminator Loss $\mathcal{L}_D$. This process employs randomness as the input for the Generator G, meanwhile, the Discriminator D endeavors to detect the authenticity of genuine data and the fidelity of the outputs produced by the Generator G. Details would be discussed in subsection 3.2.6.

1.3 What are Generative Adversarial Networks

The field of GANs has seen extensive research and development since its inception, with various architectures, improvements, applications, and challenges. The seminal work of [12] introduced GANs, as visualized in Figure 1.1, setting the foundation for a wealth of subsequent research. [13] and [14] later contributed significant improvements in training techniques and stability, giving rise to a new era of generative models. A range of architectural innovations have surfaced, including the Deep Convolutional GAN [14], Wasserstein GAN [15] for stable training, and StyleGAN [16] which revolutionized the quality of generated images. Techniques such as Spectral Normalization [17] have been instrumental in achieving further stability. GANs have found applications across diverse fields, including image synthesis [18], super-resolution [19], and data augmentation [20]. Their adaptability has facilitated novel applications, expanding the horizons of generative models. Despite progress, challenges in training stability persist [21]. Ethical concerns, specially regarding deepfakes and media manipulation, and the lack of standardized evaluation metrics [22], continue to be areas demanding further attention. The field is marked by a rapid growth of complex models like BigGAN [23] and StyleGAN2 [24], necessitating ongoing exploration of stability, interpretability, ethics, and applications. This rich landscape of research sets the context for our proposed GAN-based model, aiming to address specific challenges and contribute to this vibrant domain.

1.4 Our Proposed GAN-GPTZero and Dissertation Structure

This dissertation provides a novel Generative Adversarial Networks [12] (abbreviated as GAN) architecture for detecting GPT-generated text, dubbed as GAN-GPTZero, with a particular focus on the combination of zero-shot learning and classification training, showcasing the efficacy in the specific application area.

The structure of this dissertation is organized as follows:

- Chapter 2 discusses the motivation behind detecting GPT-generated text, followed by an explanation of the different methods that have been proposed as well as the details of GAN-based zero-shot learning and how it can be applied to GPT detection.

- Chapter 3 explores the structural design of our proposed GAN-GPTZero, presenting its architecture through mathematical formulations.

- Chapter 4 presents experiments using GAN-GPTzero and compares its performance with other state-of-the-art methods and shed new light on the mechanism under the hood.

- Chapter 5 summarizes the contributions and limitations of our approach, followed by outlining directions for future improvement.

Our contributions not only push the boundaries of what is achievable with the combination of GANs and large language models but also offer insights that may guide future research in this fascinating and rapidly evolving field.

Chapter 2

Literature Review

This chapter provides a comprehensive analysis of the existing knowledge related to GAN-GPTZero. It encompasses an extensive survey of seminal works, contemporary theories, and pertinent research findings. This chapter unfolds in the following manner: initially, the fundamental structures of GAN and LLM are delineated; subsequently, a deep exploration of the underlying theories of LLM is conducted; finally, a critical examination of the current and emergent language detectors is undertaken. The synthesis of prior research serves as the foundation upon which the present study builds.

2.1 Large Language Models With Transformers

The emergence of the Transformer architecture markes a significant milestone in NLP. Introduced by [5]. The Transformer model replaces recurrent layers with self-attention mechanisms, as depicted in Figure 2.1, enabling parallel processing and substantial efficiency gains. Following the introduction of Transformers, large-scale pretraining becomes a central theme. Models like BERT [7] and GPT-2 [1] leverages massive amounts of data and computational resources to achieve state-of-the-art performance across various NLP tasks. The success of early Transformer models leads to a plethora of variations and extensions. The

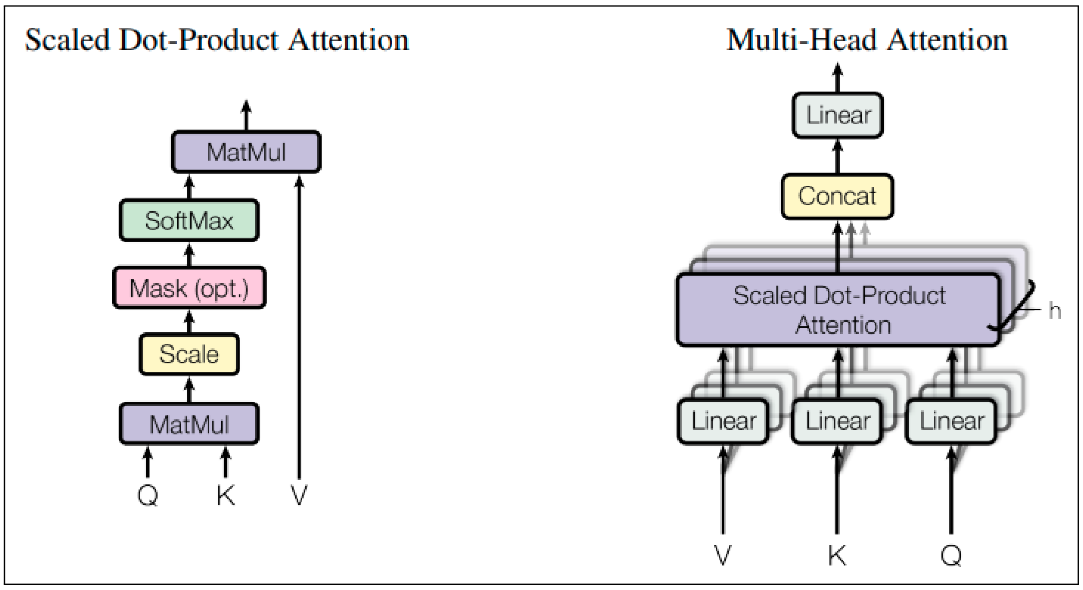


Figure 2.1: Single- & Multi-head Attention Mechanism
This picture is from [5], the left picture indicates the scaled dot-product attention, the right picture represents the multi-head attention of several attention layers running in parallel.

scaling of Transformer models reached new heights with models such as GPT-3 [3] with 175 billion parameters. Such models demonstrate a remarkable ability to generalize across tasks without task-specific tuning, moving towards more flexible and capable language understanding systems. With the growing size of Transformer models, efficiency in training and inference becomes crucial. Techniques like model parallelism [25], model distillation [26], and pruning [27] are employed to mitigate the computational challenges. The large-scale nature of these models also brings forth ethical and societal challenges, including concerns about bias [28], and environmental impact [29]. Research in multilingual models and cross-domain applications extends the reach of Transformers to diverse languages and specialized domains, bridging gaps and fostering inclusivity. From the inception of the Transformer architecture to the scaling of models and the intricate challenges of efficiency, ethics, and inclusivity, this landscape has fundamentally reshaped modern NLP. The present study builds upon these foundational works, aiming to contribute novel insights and methodologies to this dynamic research frontier.

2.2 The Mechanism inside Large Language Models

The foundational principles of large language models hinge on certain mechanisms. In recent times, two predominant mechanisms have emerged: Recurrent Neural Networks [30] and Attention. The former manifests with inherent impediments termed as vanishing gradients and exploding gradients, yet have fortunately been mitigated with the advent of the Attention mechanism. This section endeavors to delve deeply into the theoretical underpinnings of these innovative architectures.

2.2.1 RNN-based Models

Recurrent Neural Networks (abbreviated as RNN) are a class of neural networks designed to handle sequential data, making them particularly suited for tasks like text generation. The fundamental idea behind RNNs is to maintain a hidden state that gets updated at each time step of a sequence. The basic update formula for vanilla RNN is:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t) + b_h \tag{2.1}$$

where, h_t is the hidden state and x_t is the input both at time t, W_{hh} and W_{xh} are weight matrices, σ is an activation function, often the hyperbolic tangent.

While RNNs initially showed promise in text generation due to their ability to handle sequential data, they have significant limitations. Chief among these is the vanishing gradient problem, where long-term dependencies in sequences become hard to capture, causing the network to forget earlier information. Though with the advent of the other variants, including Long Short-term Memory [30] (abbreviated as LSTM) and the streamlined Gated Recurrent Units [31] (abbreviated as GRU)[1], have alleviated these problems, they still result in loss of context in longer sentences, leading to incoherent or contextually irrelevant outputs. Additionally, RNNs are inherently sequential, making parallel processing challenging and resulting in slower training times compared to other architectures.

2.2.2 Attention-based Models

The heart of the Transformer architecture lies in the Multi-Head Attention mechanism,as depicted in Figure 2.1, allowing the model to focus on different portions of the input simultaneously, capturing various types of relationships in the data. This not only solves the gradient problems but also allows for parallel computation, enhancing both speed and accuracy. The key intuition behind Multi-Head Attention is to run the attention mechanism in parallel, multiple times, allowing the model to focus on different parts of the input for each run.

For a single attention head, the mechanism can be described using the query Q, key K, and value V matrices:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{2.2}$$

where, d_k is the dimension of the keys.

The division by d_k is a scaling factor that was found to improve training stability. And for Multi-Head Attention mechanism[2], the outputs from each head are then concatenated and linearly transformed to produce the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.3}$$

[1]Refer to Appendix A for a more detailed explanation on the LSTM and GRU.

[2]Refer to Appendix B for a more detailed explanation on the Multi-Head Attention mechanism.

where, head_i is the output of the i^{th} attention head and W^O is the output weight matrix.

Through this parallel approach, the model can capture a richer combination of features and relationships, which is pivotal for understanding complex sequences like human language.

2.3 Other Proposed Generated Text Detectors

The detection mechanism has two distinct branches. The first branch of the detection strategy employs a binary classification method. As the publication of GPT2 [1], OpenAI fine-tuned RoBERTa-based [8] GPT2 detector models to differentiate texts. And also, [4] proposed DetectGPT argues that text generated from AI modes have a higher possibility to stay in negative curvature of log probability. The second branch indicates that incorporating a watermark simplifies the process of identifying LLM-generated text. As discussed in [32], the author utilizes the green and red category to imprint the special patterns. Furthermore, Appendix C delves into a more profound discussion concerning the feasibility of the detection.

2.3.1 DetectGPT Detector and RoBERTa Detector

DetectGPT [4] aims to determine if a piece of text was generated by a specific LLM. The core idea is that text sampled from an LLM tends to occupy negative curvature regions of the model's log probability function. In simpler terms, machine generated text has a distinct probability structure that can be identified by examining its ”curvature” in a probability space. Figure 2.2 provides an instinctive visualization.

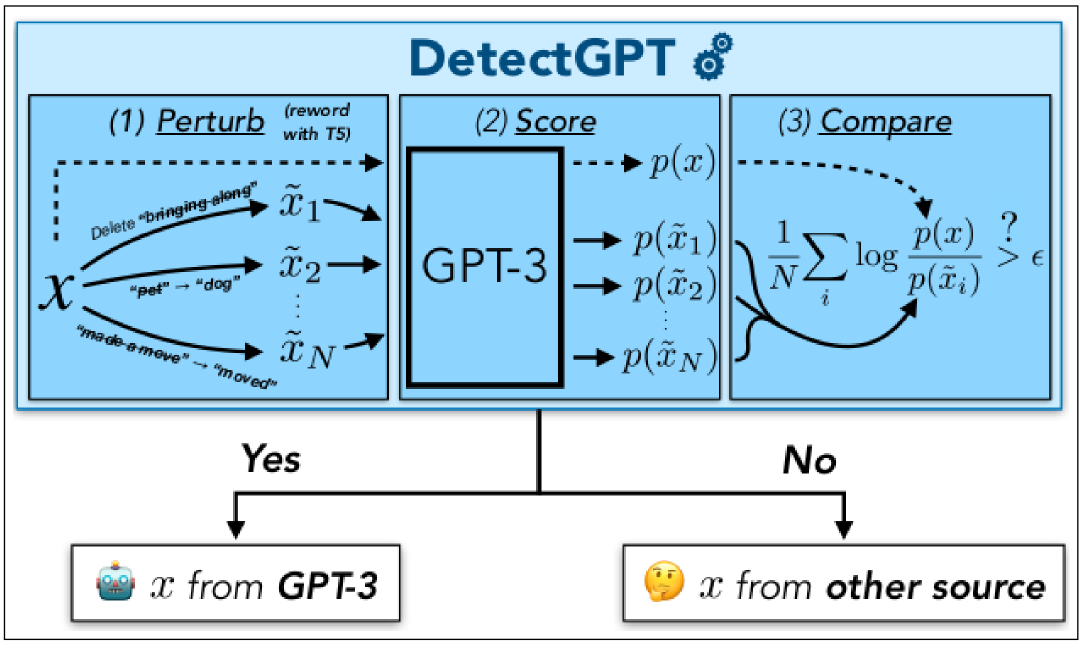


Figure 2.2: Architecture of DetectGPT
The figure from [4] elucidates the operational mechanism of the model. By introducing carefully engineered perturbations to the input text, the discriminator gains the capability to identify the model-generated content. This is accomplished through the evaluation of negative curvature present in the Negative Log-Likelihood (abbreviated as NLL) score.

The paper hypothesizes that minor perturbations of model generated text tend to have a lower log probability under the model than the original sample.[3] This can be compared by the formula:

$$d(s, p_\theta, q) = \log p_\theta(s) - E_{\tilde{s}\sim q(\cdot|s)\,\log p_\theta(\tilde{s})} \tag{2.4}$$

where, s is the original sentence, $\tilde{s}$ is the perturbed version, p_θ is the probability function of the LLM, q is the perturbation function that produces slightly modified versions of s.

In cases where q generates samples conforming to the data manifold, $d(s, p_\theta, q)$ is highly likely to be both positive and substantial for samples x emanating from p_θ. Conversely, for human written text, $d(s, p_\theta, q)$ tends to approach zero across all s.

[3]Refer to Appendix D for a more detailed explanation on the DetectGPT.

RoBERTa is a methodology introduced concurrently with the publication of GPT2. This approach entails the preparation of a dataset generated by a pre-trained GPT2 model. Subsequently, this generated dataset is merged with a corpus of human written texts to serve as the input for the RoBERTa model. The ensuing training employs a binary classification paradigm to optimize the model.

Despite the robust theoretical underpinnings of both the DetectGPT and RoBERTa detector methodologies, it is evident that there is a pronounced lack of exploration and exploitation of the collaborative dynamics between the generator and the discriminator. It is imperative to acknowledge the intrinsic 'curvature' characteristic associated with the generator: neglecting to train the generator in tandem with the discriminator could inadvertently cause a bias within the discriminator, particularly towards detecting patterns synonymous with 'curvature'. Consequently, throughout our empirical investigations, it is observed that the RoBERTa detector exhibits limited proficiency in texts that transcended its pre-existing knowledge base. Similarly, the DetectGPT framework does not align with the performance benchmarks stipulated in the original publication, indicating a potential area for further refinement and enhancement.

2.3.2 Watermark Detector

In recent publications, multiple research institutions have posited the potential of embedding a distinctive watermark[4] within generated text sequences to enhance the confidence in their detectability.

Maryland watermark [32] partitions of the vocabulary into what they term as the "green list" and "red list." Their findings suggest that such a partition makes it exceedingly challenging for an uninformed adversarial attack to circumvent the detection mechanism.

[4]Refer to Appendix E for a more detailed explanation on Watermark Strategies.

Concurrently, Stanford watermark [33] underscores their novel discovery. They assert the integration of a proprietary random sequence into the softmax [34] function not only renders the generated text identifiable but also ensures that the quality of the output remains uncompromised.

Amherst retrieval watermark [35] demonstrates that existing detectors are vulnerable to paraphrase attacks. The authors propose a retrieval-based method that searches a database of sequences previously generated by the API based on a given candidate text, looking for sequences that match the candidate text semantically.

Recently published watermark methods exhibit substantial potential, albeit their applicability remains somewhat restricted. This limitation stems from the foundational assumption that the text subjected to detection is generated with embedded watermarks, posing challenges for broader domain extension. Nevertheless, an analytical evaluation of the performance efficacy of watermark techniques will still be undertaken in subsection 4.4.3 Figure 4.3.

2.4 Advantages of GAN-GPTZero

As delineated in previous sections, the GAN-GPTZero merges the advantages inherent to both systems. This fusion integrates the ’curvature’ concept in DetectGPT and the binary classification training in RoBERTa to augment the diversity of ’curvature’ within the generator. Concurrently, this strategy enhances the discriminator’s proficiency in identifying a more heterogeneous range of ’curvature’ patterns, fostering a dynamic and adaptive learning environment.

Chapter 3

Methodology

This chapter provides an in-depth examination of the architecture and underlying theory of GAN-GPTZero. Initially, the detection problem is divided into two primary categories: 'Black Box' and 'White Box,' with each being scrutinized for its feasibility of implementation. Following this, the focus narrows to the 'White Box' category, within which GAN-GPTZero manifests in two distinct variants. The first employs a DetectGPT style methodology, while the second adopts a Roberta style approach for its discriminative component. Each of these methodologies is elucidated in detail, highlighting their specific functional roles within the broader framework

3.1 Introduction to Black Box and White Box

3.1.1 The Limitations in the Black Box Problem

It is imperative to acknowledge that the perfect emulation of human language distribution by language models remains an unattainable ideal. Different language models exhibit unique distributional characteristics, attributable to variations in architecture, training strategies, and initialization procedures. The challenge is further exacerbated by the complexity of accurately formulating the distribution of human written text, which itself is an elusive target. Moreover,

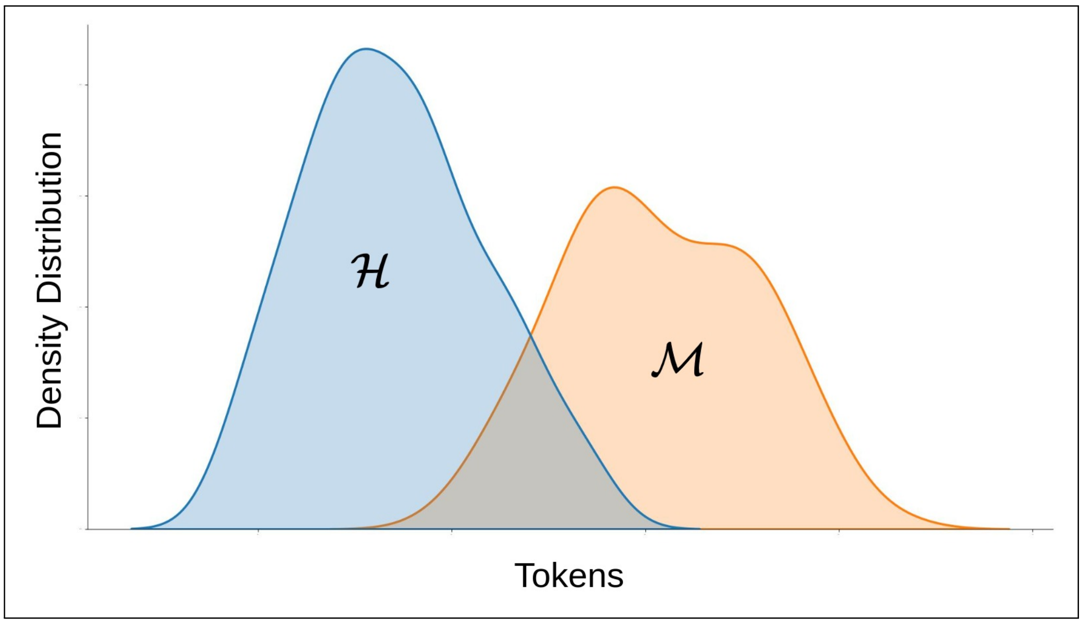


Figure 3.1: Machine Generated and Human Written Text Distribution. The horizontal axis delineates the representation of token codes, while the vertical axis illustrates the approximative distribution of the frequency histogram. The plot elucidates the inherent challenges when both human text distribution $\mathcal{H}$ and machine text distribution $\mathcal{M}$ are indeterminate, leading to subsequent mathematical computations infeasible, including the calculation of TPR_γ, FPR_γ and AUROC.

language models essentially serve as their own representations of text distribution. In scenarios where end-users lack access to the underlying model distribution and decoding strategies, it becomes theoretically infeasible to definitively ascertain the provenance of a given text sample. The paper [36] furnishes the following mathematical evidence:

Let $\mathcal{H}$ and $\mathcal{M}$ denote the human and machine generated text distributions, respectively. Given the Discriminator D processes the input sentence s and outputs a confidence score, a probability measure $\mathtt{P}$, then the True Positive Rate (TPR_γ) and the False Positive Rate (FPR_γ) under a specified threshold γ are articulated as:

$$\text{TPR}_\gamma = \mathrm{P}_{s \in \mathcal{M}}(D(s) \geq \gamma) \tag{3.1}$$

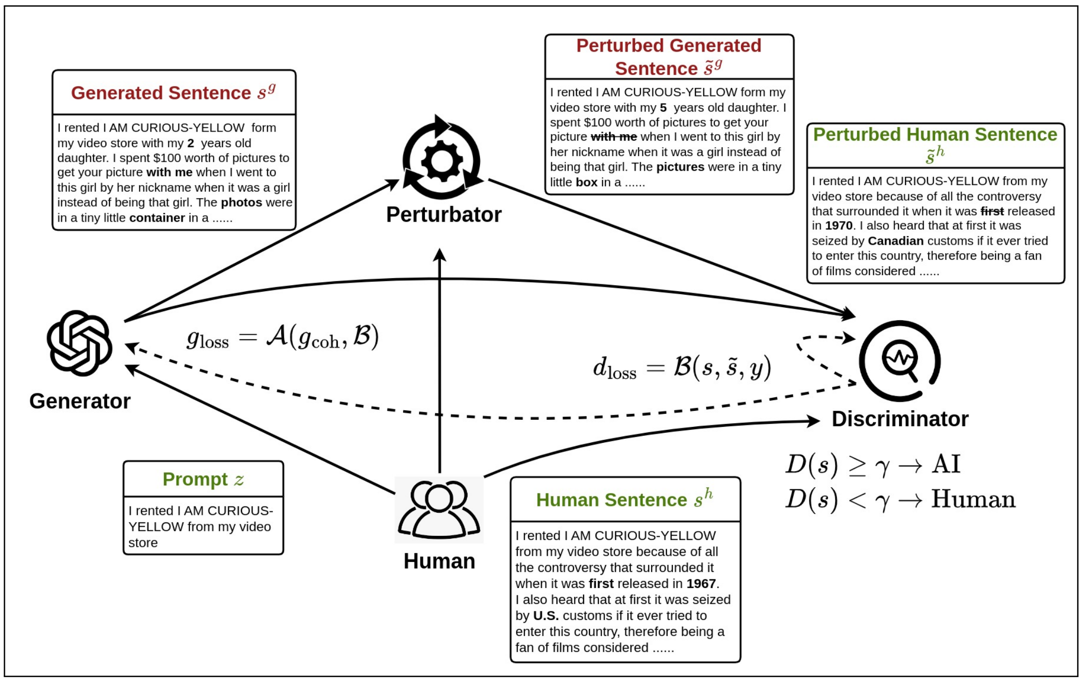
$$\text{FPR}_\gamma = \mathrm{P}_{s \in \mathcal{H}}(D(s) \geq \gamma) \tag{3.2}$$

$$\text{AUROC} = \int (\text{TPR}_\gamma) d(\text{FPR}_\gamma) \tag{3.3}$$

The key of our argument lies in the unavailability of the exact knowledge regarding $\mathcal{M}$, though $\mathcal{H}$ still could be approximated using the empirical distribution derived from the dataset. Due to the intrinsic limitation, computing the Area Under the Receiver Operating Characteristic Curve (abbreviated as AUROC), a widely-accepted metric for binary classification, becomes theoretically untenable, a more intuitive expression is shown in Figure 3.1.

3.1.2 The Feasibility of the White Box Problem

In the domain of white box models, wherein the architecture is publicly accessible, several foundational models serve as the backbone—these include GPT, BERT, OPT [37], LLAMA [38], LLAMA2 [39] and so forth. Notably, while GPT3 and GPT4 [40] may not be openly accessible, the open-source availability of GPT2 can still offer valuable insights. This is primarily because the primary differentiation among various GPT versions lies in their scales, rather than their fundamental architectures. In such scenarios, the machine text distribution $\mathcal{M}$ becomes available. In practice, an approximation $\hat{\mathcal{M}}$ is often used. It can be rigorously demonstrated that during self-detection—wherein the generative model itself serves as the backbone for the discriminator—the approximation holds as $\hat{\mathcal{M}} \approx \mathcal{M}$. Furthermore, the Equation 3.2 can be readily computed using empirical dataset. Consequently, through this methodology, both Equation 3.1 and Equation 3.2 become attainable, and Equation 3.3 sufficiently furnishes the comparative analysis results.


Generated Sentence s^g
Perturbed Generated Sentence $\tilde{s}^g$
Perturbed Human Sentence $\tilde{s}^h$
Perturbator
Generator
$g_{\text{loss}} = \mathcal{A}(g_{\text{coh}}, \mathcal{B})$
$d_{\text{loss}} = \mathcal{B}(s, \tilde{s}, y)$
Discriminator
$D(s) \geq \gamma \rightarrow \text{AI}$
$D(s) < \gamma \rightarrow \text{Human}$
Prompt z
Human
Human Sentence s^h

I rented I AM CURIOUS-YELLOW form my video store with my **2** years old daughter. I spent $100 worth of pictures to get your picture **with me** when I went to this girl by her nickname when it was a girl instead of being that girl. The **photos** were in a tiny little **container** in a ......

I rented I AM CURIOUS-YELLOW form my video store with my **5** years old daughter. I spent $100 worth of pictures to get your picture ~~**with me**~~ when I went to this girl by her nickname when it was a girl instead of being that girl. The **pictures** were in a tiny little **box** in a ......

I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was ~~**first**~~ released in **1970**. I also heard that at first it was seized by **Canadian** customs if it ever tried to enter this country, therefore being a fan of films considered ......

I rented I AM CURIOUS-YELLOW from my video store

I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was **first** released in **1967**. I also heard that at first it was seized by **U.S.** customs if it ever tried to enter this country, therefore being a fan of films considered ......

Figure 3.2: Workflow of GAN-GPTZero DetectGPT Style Architecture
The diagram presents the comprehensive architecture of GAN-GPTZero, incorporating a DetectGPT style methodology. This architecture consists of three core components: the Generator G, the Perturbator P, and the Discriminator D. A dataset containing human written sentence s^h serves as the source of prompts z which is simultaneously fed into the Generator G. Concurrently, the Perturbator P receives both the generated sentence s^g from the Generator G and the original human written sentence s^h. Subsequently, the Discriminator D computes the discriminator loss $\mathcal{B}(\cdot)$ (Equation 3.29) based on perturbed $\tilde{s} = \{\tilde{s}^g, \tilde{s}^h\}$ as well as original $s = \{s^g, s^h\}$. This calculated discriminator loss not only serves to update the Discriminator D but also contributes to the iterative refinement of the Generator G by the designed loss function $\mathcal{A}(\cdot)$ (Equation 3.35), wherein the loss function g_{coh} (Equation 3.14) is calculated from the generation and would be delineated in subsection 3.2.3

3.2 GAN-GPTZero DetectGPT Style Architecture

Utilizing DetectGPT as the discriminative component, the GAN-GPTZero architecture comprises three key modules: the Generator G, the Perturbator P, and the Discriminator D, as illustrated in Figure 3.2. Within this framework, the Perturbator P serves the specific function of introducing controlled disturbances into the generated sentence s^g. Concurrently, the Discriminator D is trained in tandem with the Generator G, guided by the loss function $\mathcal{B}(\cdot)$ and $\mathcal{A}(\cdot)$

specific to Generative Adversarial Networks.

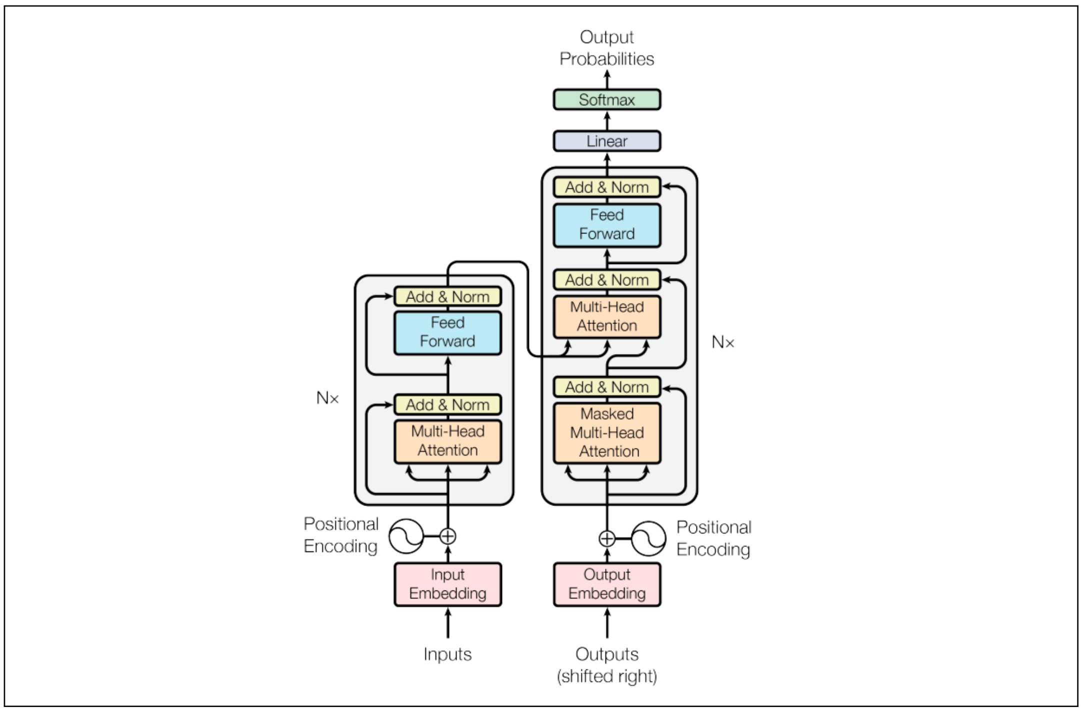


Figure 3.3: Transformer Architecture
This figure, sourced from [5], vividly portrays the architectural composition of Transformers, mainly employing layered attention mechanisms. The left branch functions as the encoder, tasked with encrypting and learning the intricacies of semantic connections, while the right branch operates as the decoder, facilitating the generation of content derived from encrypted semantic information.

3.2.1 Generation

Two primary rationales underlie the selection of GPT models as the foundational architecture for the Generator G are as follows. The initial reason pertains to that a distinctive characteristic of GPT is its use of masked self-attention in the Transformer as shown in Figure 3.3, ensuring each token can only attend to previous tokens, maintaining the autoregressive property essential for generative tasks. The secondary justification centers on the fact that GPT could be finetuned on specific tasks by simply adding a linear output layer as illustrated in [2], training on labeled data, and minimizing the corresponding loss. The versatility and efficacy of GPT have made it a cornerstone in modern NLP, with subsequent iterations further advancing state-of-the-art benchmarks across a myriad of tasks.

Rooted in the Transformer architecture championed by [5], GPT showcases a two-step paradigm: pretraining on a vast corpus to learn a language model and finetuning on specific tasks. The GPT model's architecture is primarily composed of stacked transformer decoders, as the decoder visualized in Figure 3.3. Given an input sequence s of tokens $x_1, x_2 \ldots, x_n$, the objective in both the pretraining and finetuning phases is to maximize the likelihood of predicting the next token $\hat{x}_{t+1}$. Formally, this is represented as:

$$L_G(\theta_g) = \sum_t \log \mathbb{P}(\hat{x}_{t+1} | x_1, x_2, \ldots, x_t; \theta_g) \tag{3.4}$$

where, θ_g represents the parameters of Generator G, $L_G(\theta_g)$ is the likelihood to be maximized, and the $\mathbb{P}$ denotes a predicted probability distribution over conditions.

In the framework of this study, it is pivotal to note that the Perturbator P is configured to process texts as inputs, as opposed to logits. This necessitates the text to be generated from the underlying generative model. As has been elaborated earlier, the choice of sampling and decoding strategies is not trivial and must be made with care. In this particular methodology, the nucleus sampling [41] is employed as the text generation technique.

3.2.2 Nucleus Sampling

Instead of sampling only from the most likely K words as indicated in Top-k sampling [1] [42], Top-p sampling [41][5] chooses from the smallest possible

[5]Refer to Appendix F for a more detailed explanation on decoding strategies.

set of words whose cumulative probability exceeds the threshold probability p. The probability mass is then redistributed among this set of words. In this way, the size of the set of words N (a.k.a the number of words in the set) can dynamically increase and decrease according to the next word's probability distribution.

Mathematically, with the next token candidate $\hat{x}'_{t+1}$ at the time step t from a given vocabulary V, the nucleus set N is defined as:

$$N = \{\hat{x}'_{t+1} \in V \mid \mathbb{P}(\hat{x}'_{t+1} \mid x_{1:t}) \geq p\} \tag{3.5}$$

where, p is a predefined threshold, typically in the range $(0, 1)$.

Once N is established, the ensuing token x_{t+1} is sampled from N in the following manner:

$$\hat{x}_{t+1} = \arg\max_{\hat{x}_{t+1}} \mathbb{P}(\hat{x}'_{t+1} \mid \hat{x}'_{t+1} \in N; x_{1:t}) \tag{3.6}$$

where, $\mathbb{P}(\hat{x}'_{t+1} \mid \hat{x}'_{t+1} \in N, x_{1:t})$ is the conditional probability associated with the previous tokens $x_{1:t}$ at time step t in Equation 3.4 conditioned upon its presence in the nucleus set N. Essentially, it also serves as the normalized probability of $\hat{x}_{t+1}$ within the confidences of N.

Furthermore, the prompt z from the human written sentence s^h is mathematically defined as:

$$z = \mathcal{Z}(s^h) \tag{3.7}$$

update to the batch level:

$$Z = \mathcal{Z}(S^h) \tag{3.8}$$

where, $\mathcal{Z}$ is a dataset building function which would be further illustrated in subsection 4.1.2, S^h represents a batch of s^h, with a batch size of m.

Therefore, the generated sentence s^g can be mathematically formulated as:

$$s^g \triangleq \text{Concat}\{z,\ (\arg\max_{\hat{x}_{t+1}} \sum_t \mathbb{P}(\hat{x}'_{t+1} \mid \hat{x}'_{t+1} \in N;\ \text{Concat}\{z, x_{1:t}\}))\} \tag{3.9}$$

where, the operation Concat$\{\cdot\}$ signifies the concatenation of the first and the second entities in the expression.

yielding:

$$s^g = \{z, s^g_1, s^g_2, \ldots, s^g_n\} = \{z, \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\} \tag{3.10}$$

for simplification and in accordance with Equation 3.7, Equation 3.9 is expressed as:

$$s^g = G(\mathcal{Z}(s^h)) \tag{3.11}$$

While considering the batch level Equation 3.8, the Equation 3.11 is updated as:

$$S^g = G(\mathcal{Z}(S^h)) \tag{3.12}$$

The advantages of nucleus sampling includes its dynamic adaptability, allowing it to potentially capture rare but contextually relevant words. Moreover, it avoids the arbitrariness of selecting a fixed K value, making it more flexible in diverse contexts. However, similar to other sampling methods, the choice of p is crucial. A high p could lead to more randomness, while a low p might make the output more deterministic. Proper tuning of this hyperparameter is essential to ensure a balance between diversity and contextual relevance in generated sequences.

3.2.3 Next Word Cross-Entropy Loss

In models built upon the GPT2 architecture, the task of predicting the subsequent word in a sequence is formulated as a multi-class classification problem. Let x_{t+1} represent the ground-truth token at the time step t in a given sentence, which comes from the human written sentence s^h, and $\hat{x}_{t+1}$ is the model's predicted probability distribution over the vocabulary V at that same time step, as shown in Equation 3.4. The loss function at each time step t is based on the Cross-Entropy Loss between $\hat{x}_{t+1}$ and x_{t+1}.

The Cross-Entropy Loss for an individual time step t is defined as follows:

$$\text{CE}(x_{t+1}, \hat{x}_{x+1}) = \sum x_{t+1} \log \hat{x}_{t+1} \tag{3.13}$$

next, the average next word Cross-Entropy Loss g_{coh} across the entire texts batch is given by:

$$g_{\text{coh}}(S^h) = -\frac{1}{mn} \sum_{i}^{m} \sum_{t}^{n} S_{it}^h \log S_{it}^g \tag{3.14}$$

where, S_{it}^g is the generated token at the time step t in the i^{th} sentence in a batch and g_{coh} serves as the loss function employed during the training of the Generator G . The term "coh", refers to "coherence", signifies that the next word Cross-Entropy Loss serves as a metric to gauge the coherence within the generated sentence s^g.

3.2.4 Perturbation

The T5 model, which is a sequence to sequence model based on the architecture of BERT, is selected as the Perturbator P. It adds noise to the text by changes on the word-level without too much deviations on meaning of the text. The BERT [7] marks a significant shift in the way text representations are learned. Contrary to models that read text sequences in one direction (either left-to-right or right-to-left), BERT is designed to read in both directions, offering a deeper understanding of context.

The core innovation in BERT is its bidirectional attention mechanism, which leverages the Transformer's architecture, specifically its encoder stacks as shown in Figure 3.3. The primary pretraining method involves masking a fraction of the input tokens and predicting them, formally known as the Masked Language Model (abbreviated as MLM) objective. Given an input sequence s of tokens $x_1, x_2, \ldots x_n$, where a subset of tokens x^* are masked, the objective is to predict the masked tokens:

$$L_P(\theta_p) = -\log \mathcal{P}(x^* | x_1, x_2, \ldots, x_t; \theta_p) \tag{3.15}$$

where, θ_p represents the parameters of Perturbator P, $L_P(\theta_p)$ is the likelihood to be maximized, and the $\mathcal{P}$ denotes a predicted probability distribution over conditions.

As this training method, the T5 models provide predicting, deleting and swapping functions. Suppose the ζ contains the randomness and masking information, the θ_p is the finetuned parameters of T5 model, the perturbed output $\tilde{s}$ is formulated as:

$$\tilde{s} = \mathcal{P}(x^* \mid s; \theta_p, \zeta) \tag{3.16}$$

As a consequence, given machine generated sentence s^g and human written sentence s^h, the Perturbator P yields the following outputs:

$$\tilde{s}^g = P(s^g) \tag{3.17}$$

$$\tilde{s}^h = P(s^h) \tag{3.18}$$

where, $\tilde{s}^g$ represents the perturbed machine generated text, and the $\tilde{s}^p$ represents the perturbed human written text.

formulate Equation 3.17 and Equation 3.18 to the batch level as:

$$\tilde{S} = \{\tilde{S}^g, \tilde{S}^h\} = P(S) \tag{3.19}$$

where, the S is consisted of $\{S^g, S^h\}$.

It is noteworthy that the perturbed texts $\tilde{S}^g$ and $\tilde{S}^h$ maintain a high degree of semantic congruence with their respective original texts S^g and S^h.

3.2.5 Discrimination

The focal point of this discussion lies in the Discriminator D. Both [43] and [44] posit that smaller models within the same architectural family exhibit a heightened sensitivity to discrepancies between human written and machine generated text distributions. Consequently, GPT2 and GPT2-Medium are selected as the Discriminator D for GAN-GPTZero architecture.

The Discriminator D employs a Binary-Entropy Loss function, denoted as d_{loss}. Given the perturbed input sentence $\tilde{s} = \{\tilde{s}^g, \tilde{s}^h\}$ and original input $s = \{s^g, s^h\}$, the Discriminator D processes a mixed input $\{s, \tilde{s}\}$ then predicts confidence score f based on the generation $\hat{s} = \{\hat{s}^g, \hat{s}^h\}$ from itself, representing input is classified as machine generated text f^g_γ or human written text f^h_γ under a threshold γ, respectively. The process of Discriminator D is formulated as follows:

for both s and $\tilde{s}$, the generation $\hat{s}$ is defined similar from Equation 3.9:

$$\hat{s} = \sum_{t}^{n} \arg\max_{\hat{x}_{t+1}} \mathtt{P}(\hat{x}_{t+1} \mid x_{1:t}; \theta_d) \tag{3.20}$$

where, x represents token in s as well as $\tilde{s}$, $\mathtt{P}$ denotes the predicted probability distribution over the parameters of Discriminator D, denoted as θ_d.

from Equation 3.13 and Equation 3.20, the Negative Log-Likelihood Loss f_s of input s is defined as:

$$f_s(s) = -\frac{1}{n} \sum_{t}^{n} s_t \log \hat{s}_t \tag{3.21}$$

similar, the Negative Log-Likelihood Loss $f_{\tilde{s}}$ of input $\tilde{s}$ is defined as:

$$f_{\tilde{s}}(\tilde{s}) = -\frac{1}{n} \sum_{t}^{n} \tilde{s}_t \log \hat{s}_t \tag{3.22}$$

then the confidence score is thus defined as:

$$f = \frac{f_s - f_{\tilde{s}}}{f_{\tilde{s}}} \tag{3.23}$$

the prediction under the threshold γ is thus formulated as:

$$f \triangleq \{f^g, f^h\}: \begin{cases} f^g = 1, & \text{if} \quad f \geq \gamma \\ f^h = 0, & \text{else} \end{cases} \tag{3.24}$$

integrate Equation 3.20, Equation 3.23 and Equation 3.24 as:

$$f = D_\gamma(s, \tilde{s}) \tag{3.25}$$

Further suppose the actual labels y of machine generated and human written are $y^g = 1$ and $y^h = 0$, respectively. Considering Equation 3.23 and Equation 3.24, the discriminator loss function d_{loss} for binary classification is subsequently defined as:

$$d_{\text{loss}}(f, y) = -y \log f \tag{3.26}$$

update both Equation 3.25 and Equation 3.26 to the batch level as:

$$F = D_\gamma(S, \tilde{S}) \tag{3.27}$$

$$d_{\text{loss}}(F, Y) = -\sum_{i}^{m} Y_i \log F_i \tag{3.28}$$

where, F represents a batch of predictions f, Y represents a batch of labels y, both of them are with a batch size of m.

with a condensed formulation $\mathcal{B}(\cdot)$, Equation 3.28 could be succinctly expressed as:

$$d_{\text{loss}} \triangleq \mathcal{B}(S, \tilde{S}, Y) \tag{3.29}$$

This discriminator loss function d_{loss} serves dual purposes: it updates the Discriminator D and also influences the Generator G. However, the presence of the Perturbator P as well the attribute of GAN-GPTZero architecture, which involves a decoding step that disrupts the computational graph, hinders the use of the conventional GAN loss. An alternative approach to this issue is explored in the subsequent subsection 3.2.6.

3.2.6 Meaning Loss

In conventional GAN architectures, the Generator G and Discriminator D are jointly trained using generation loss function $\mathcal{L}_G$ and discrimination loss function $\mathcal{L}_D$, often derived from Binary Cross-Entropy Loss. Specifically, with the given human written texts S^h, the standard GAN loss $\mathcal{L}_{\text{GAN}}$ can be articulated as:

$$\mathcal{L}_{\mathrm{G}} = \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathcal{Z}(S_i^h)))) \tag{3.30}$$

$$\mathcal{L}_{\mathrm{D}} = \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(S_i^h) + \log(1 - D(G(\mathcal{Z}(S_i^h))] \tag{3.31}$$

$$\begin{aligned}\min_G \max_D \mathcal{L}_{\mathrm{GAN}}(\mathrm{G}, \mathrm{D}) = &\mathbb{E}_{S^h \sim p^{\mathrm{h}}(S^h)}[\log(D(S^h))] + \\ &\mathbb{E}_{Z \sim p^{\mathrm{p}}(Z)}[(\log(1 - D(G(\mathcal{Z}(S^h)))))]\end{aligned} \tag{3.32}$$

where, $\mathbb{E}$ signifies expectation, p^{h} is the distribution of human written text S^h, and p^{p} is the prompt distribution derived from the Equation 3.8, m is the batch size, ∇ is the gradient of the chosen variables, θ_g and θ_d stand for the Generator G and Discriminator D, respectively.

However, in the GAN-GPTZero framework, the vanilla GAN loss formulations (Equation 3.30, Equation 3.31, Equation 3.32) lose its applicability. This is because G and D do not exchange hidden states but rather communicate through texts modified by a Perturbator P. To circumvent this limitation, the concept of a Meaning Loss g_{mng} is introduced.

Critically, the traditional g_{coh} only directs G towards next word prediction and lacks the capacity to guide towards generating more human-like or machine-like text. This guidance is predominantly influenced by d_{loss}. To integrate feedback from Discriminator D back into Generator G, the Meaning Loss g_{mng} is defined as:

$$g_{\mathrm{mng}} \triangleq d_{\mathrm{loss}} \cdot g_{\mathrm{coh}} \tag{3.33}$$

where, g_{coh} is from Equation 3.14, d_{loss} is from Equation 3.29, respectively.

Although g_{mng} incorporates information from Discriminator D, it is insufficient as a standalone loss function for G. This is primarily because, at initialization, D is often too strong, thereby diluting the fluency and coherence encapsulated by g_{coh}. To maintain a balance, an overall generator loss g_{loss} is introduced as:

$$g_{\text{loss}} \triangleq \alpha \cdot g_{\text{mng}} + (1-\alpha) \cdot g_{\text{coh}} \tag{3.34}$$

where, α is a tunable hyperparameter named focus weight. Empirical results suggest that $\alpha = 0.3$ yields optimal performance.

Given that both g_{coh} and d_{loss} are directly derived from the GAN-GPTZero architecture as well as the condensed formulation $\mathcal{A}(\cdot)$, Equation 3.34 can be succinctly expressed as:

$$g_{\text{coh}} = \mathcal{A}(g_{\text{coh}}, \mathcal{B}) \tag{3.35}$$

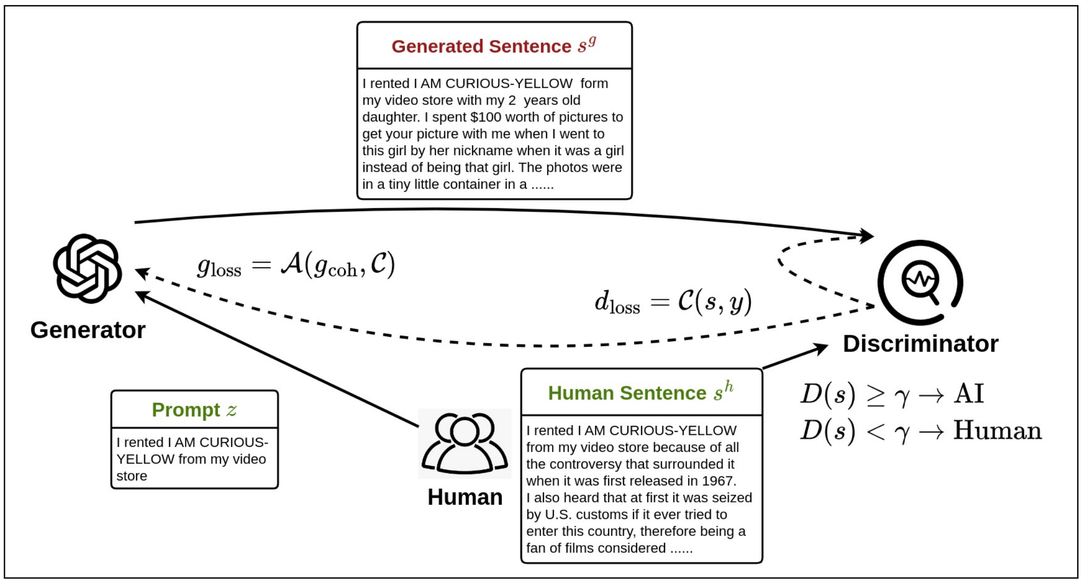

Generated Sentence s^g
I rented I AM CURIOUS-YELLOW form my video store with my 2 years old daughter. I spent $100 worth of pictures to get your picture with me when I went to this girl by her nickname when it was a girl instead of being that girl. The photos were in a tiny little container in a ......
$g_{\text{loss}} = \mathcal{A}(g_{\text{coh}}, \mathcal{C})$
$d_{\text{loss}} = \mathcal{C}(s, y)$
Generator
Discriminator
Prompt z
I rented I AM CURIOUS-YELLOW from my video store
Human
Human Sentence s^h
I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was first released in 1967. I also heard that at first it was seized by U.S. customs if it ever tried to enter this country, therefore being a fan of films considered ......
$D(s) \geq \gamma \rightarrow$ AI
$D(s) < \gamma \rightarrow$ Human

Figure 3.4: Workflow of GAN-GPTZero RoBERTa Style Architecture
The figure delineates the architecture based on the RoBERTa style method without the Perturbator P, thereby simplifying the overall structure. Despite this simplification, the Generator G and the Discriminator D employ distinct tokenizers. As a consequence, the designed generator loss $\mathcal{A}(\cdot)$ continues to be relevant.

3.3 GAN-GPTZero RoBERTa Style Architecture

Contrary to the detectgpt approach, the Roberta-based method serves as OpenAI's proprietary detector, trained explicitly on outputs from various GPT models. Our empirical evaluation indicates that it is a highly effective discriminator, largely owing to its extensive training dataset. The limited efficacy of Detectgpt can be attributed to its zero-shot nature, as it does not undergo training specifically for the task.

In this configuration, the Discriminator D is embodied by the RoBERTa model, a non-generative, masked language model that neither shares the same architecture nor employs the same tokenizer as GPT2. Consequently, although the architecture no longer includes an intermediary Perturbator P between the Generator G and Discriminator D, the necessity for the Meaning Loss g_{mng} persists. As inferred from the previous discussion, the formulations for the modified GAN loss (Equation 3.29, Equation 3.35) remain almost unchanged. The architecture underlying the Roberta style method is graphically depicted in Figure 3.4.

For conciseness, the RoBERTa style method is formulated at the batch level. Given the corpus of human written texts S^h, the prompts Z is derived from Equation 3.8 and the generated texts S^g originates from Equation 3.12. Consequently, the predictions F from the Discriminator D with the input $S = \{S^g, S^h\}$ under the threshold γ is formulated as:

$$F = \{F^g, F^h\} = D_\gamma(S) \tag{3.36}$$

yet to be noted, Equation 3.36 is appended with a classification head layer followed by a softmax [34] layer.

from Equation 3.29, wherein Y serves as labels, d_{loss} is modified as:

$$d_{\text{loss}}(S) = \mathcal{C}(\mathcal{S}) = -\sum_{i}^{m} Y_i \log F_i \tag{3.37}$$

where, $\mathcal{C}(\cdot)$ denotes the condensed formulation.

Considering Equation 3.14 and Equation 3.37, the g_{loss} is formulated as:

$$g_{\text{loss}} = \mathcal{A}(g_{\text{coh}}, \mathcal{C}) \tag{3.38}$$

Owing to the absence of the Perturbator P, the mathematical formulation of the RoBERTa style method is conciser.

3.4 Summary of Methodology

Both the GAN-GPTZero Detect Style and GAN-GPTZero RoBERTa Style architectures enhance the interaction between the Generator G and Detector D in the linguistic space, enriching their connectivity. This chapter elucidates the mathematical formulations underpinning these two architectures, thereby establishing a robust foundation for the subsequent training process.

Chapter 4

Experiment

This chapter substantiates the efficacy and serves as an empirical foundation for the claims made regarding the robustness and effectiveness of GAN-GPTZero. The structure is arranged as follows. Firstly, discuss the method used on the construction of the training and experimental dataset, along with their respective functionalities. Secondly, an analytical comparison is drawn to illustrate the superior generation capabilities of GAN-GPTZero in terms of perplexity. Thirdly, a comparative analysis scrutinizes the similarity between the model distribution and human language distribution, thereby demonstrating that GAN-GPTZero exhibits a distribution more akin to natural language patterns. Further, a critical AUROC evaluation of the discrimination accuracy of GAN-GPTZero is undertaken, compared to other state-of-the-arts, and also the evidence of the potential improvements on the detection capabilities is provided. Lastly, empirical experiments indicate that our model even surpasses human evaluation accuracy.

4.1 Preparation of Training Dataset

This section is to elaborate the methodology employed for dataset preparation, specifically designed for the intricate task of training GAN-GPTZero. According to existing literature, notably with respect to [1], augmenting the training data with a heterogeneous mixture of datasets can enhance the performance of the discriminator module. To ensure a comprehensive coverage across a broad spectrum of topics and genres, the dataset is assembled from various sources and subjected to rigorous preprocessing to maintain data integrity and quality.

4.1.1 Multi-Dataset Integration and Preprocessing

In the initial stage of dataset selection, it is noteworthy that GPT2 has already been pretrained on expansive and diverse datasets, including WebText [1],Book-Corpus [45], and CommonCrawl. Consequently, there is no imperative need to finetune GAN-GPTZero on these substantially large datasets. Instead, it opts for more specialized datasets, namely, IMDb [46], SQuAD [47] [48], and AmazonReviews. All of these datasets have been sourced from the Hugging Face datasets repository [49], ensuring their credibility and wide acceptance in the machine learning community.

In the dataset integration process, multiple steps are taken to ensure uniformity, cleanliness, and relevance. These steps include transformation, cleaning, and formation of the data.

For data transformation, the inherent structural differences among the selected datasets necessitate format standardization. Specifically, the SQuAD dataset comes in the shape of [’id’, ’title’, ’context’, ’question’, ’answers’] with a total of 98,565 text entries. The IMDb dataset has the shape [’text’, ’label’] and consists of 100,000 text entries, whereas the AmazonReviews dataset is structured as [’train’, ’test’] with a total of 3,650,000 text entries. To align these disparate formats, each dataset is uniformly transformed into a shape with a single column [’text’]. This is achieved either by discarding extraneous columns or by merging the ’train’ and ’test’ columns, as applicable.

For data cleaning, the datasets are subjected to a rigorous cleaning process to remove noise, including the presence of non-standard symbols and foreign language text. A stringent criterion is established for this cleaning step: if a text entry contains more than k anomalous symbols or foreign language words, it is deemed nonsensical and subsequently removed from the dataset. Empirical tests indicate that optimal performance is achieved when $k = 5$. Although this cleaning process may reduce the overall dataset size, it enhances the quality, ensuring that the model is less susceptible to noise and irrelevant features.

For data formation, to further enhance the model's performance, the datasets are intermixed during the training phase. The rationale behind this approach is drawn from literature that suggests diversified datasets contribute to a more robust discriminator. To control the degree of randomness in this mixture, a seed parameter is utilized.

Additionally, the literature [1] emphasizes the utility of mixing sentence lengths during training to achieve improved model performance. This is evidenced by findings from other works such as [44] and [50], indicating that longer sentences contribute to a more accurate discriminator.

However, while there is a direct relationship between sentence length and discriminator accuracy, this has to be balanced against computational capacity and utility. To strike an optimal trade-off, text entries are truncated or segmented to fit within a word count range of 128 to 256 words, approximating the length of a typical paragraph. This ensures that the model benefits from both the increased accuracy associated with longer sentences and the computational efficiency required for practical applications.

By implementing this strategy, each training batch is ensured to contain samples from all three datasets, SQuAD, IMDb and AmazonReviews. This not only broadens the range of information available to the model in each iteration but also mitigates the risk of dataset-specific biases. Consequently, the GAN-GPTZero benefits from a well-rounded and less skewed training experience.

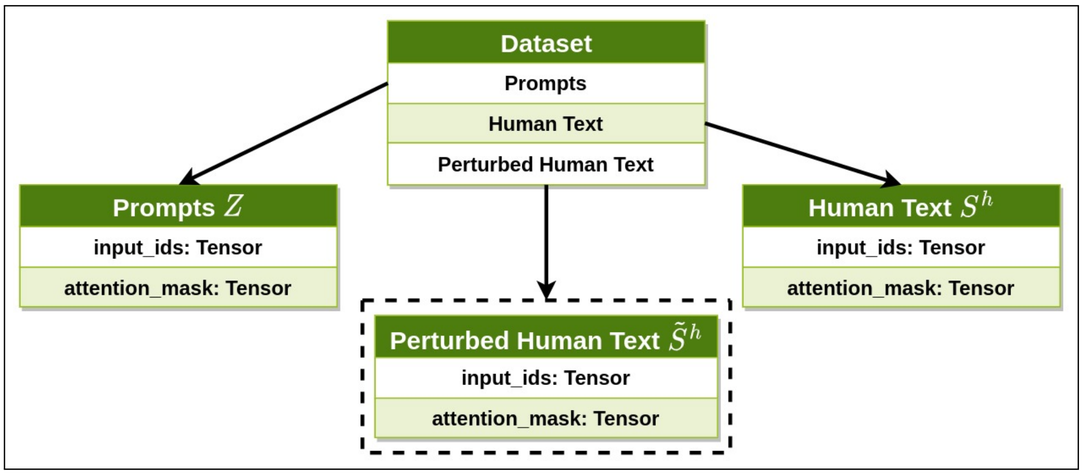


Figure 4.1: GAN-GPTZero Variants Dataset Structure.
DetectGPT style dataset is with the dashed line frame yet RoBERTa style dataset is without. Dataset consists of 'Prompts', 'Human Text', and 'Perturbed Human Text', all of which are textual in nature. The arrows signify the process of tokenization executed by the relevant tokenizers. The tokenization output conforms to the HuggingFace dataset [49] structure, specifically comprising two key elements: 'input_ids' and 'attention_mask', both of which are represented as tensors. Appendix G provides a sample of the dataset for a clearer demonstration.

4.1.2 Dataset Structure for DetectGPT Style and RoBERTa Style

According to the methodology chapter, the dataset preparation strategy was delineated for both DetectGPT Style and RoBERTa Style. Owing to their inherent structural differences, each requires a unique dataset configuration.

For the RoBERTa Style method, the dataset essentially consists of two columns: 'Prompts' and 'Human Text'. Prompts are derived by selecting the first 10 words from each entry in the preprocessed dataset, which is formulated by Equation 3.8. These prompts are then concatenated with the corresponding 'Human Text', which are the original, full-length text entries. Consequently, the dataset for the RoBERTa Style method is structured as ['Prompts', 'Human Text'].

For the DetectGPT Style method, an additional layer of complexity is introduced by the inclusion of a Perturbator P. To expedite the training process, the perturbation of 'Human Text' is performed offline. This leads to the addition of a third column in the dataset, termed 'Perturbed Human Text'. Therefore, the DetectGPT style dataset takes the shape ['Prompts', 'Human Text', 'Perturbed Human Text']. Figure 4.1 offers a comprehensive visualization that elucidates the GAN-GPTZero variants dataset structure.

4.1.3 Functions of Different Columns in Dataset

The functional roles of various columns in the datasets are tailored differently. Understanding the utility of each column is pivotal for grasping the flow of information during the training process.

In the RoBERTa style method, the dataset is structured as ['Prompts', 'Human Text']. Here, the 'Prompts' serves a dual role:

- - Input to the generator G: The 'Prompts' Z is fed into the Generator G, which in turn produces the 'Generated text' S^g.
- Input to the Discriminator D: Both the 'Generated Text' S^g and the 'Human Text' S^h are used as the inputs for the Discriminator D for prediction.

For the DetectGPT style method, the dataset structure extends to ['Prompts', 'Human Text', 'Perturbed Human Text']. The columns functions are as follows:

- Input to the Generator G: Similar to the RoBERTa Style, the 'Prompts' Z serves as the input to the Generator G, resulting in a 'Generated Text' S^g.

- Input to the Discriminator D: The 'Perturbed Generated Text' $\tilde{S}^g$ and the 'Perturbed Human Text' $\tilde{S}^h$ along with the corresponding 'Generated Text' S^g and 'Human Text' S^h are used as the inputs to the Discriminator D for prediction.

For both methodologies, once the discriminator predicts scores, the loss functions (Equation 3.14, Equation 3.29, Equation 3.35, Equation 3.37, Equation 3.38) would be computed. Utilizing the backpropagation algorithm [51], gradients are calculated and updated, thereby initiating the training process.

4.2 Preparation of Testing Dataset

The integrity of an experimental evaluation hinges critically on the dataset used. To mitigate the risk of overfitting and to ensure an unbiased assessment, it is imperative that the experimental dataset be entirely distinct from the datasets used for both pretraining and finetuning. Given the potential for overlaps between these pre-existing datasets and other commonly-used benchmarks, an opted novel approach is: formulating dataset from the BBC News website.

As of August 29, 2023, the curated dataset comprises 1,000 news excerpts, each with a length ranging between 128 and 256 number of words. This configuration is consistent with the word-length range utilized in the training datasets. After the preprocessing as depicted in subsection 4.1.2, the experimental dataset adopts a structure that includes three key components: ['Prompts',' 'Human Text', 'Perturbed Human Text'], mirroring the construction of the training dataset.

4.3 Build and Train the Model

In the implementation of the RoBERTa style method, the GPT2-Large model is chosen as the Generator G. For the Discriminator D, two variants of RoBERTa are utilized: RoBERTa-Base with 125 million parameters and RoBERTa-Large with 356 million parameters. Conversely, in the DetectGPT Style approach, the GPT2 and GPT2-Medium are adopted as the Discriminators D. It is noteworthy that in both methodologies, GPT2-Large serves as the Generator G. These model variants are named as **GR-Small**, **GR-Large**, **GD-Small**, **GD-Large**, corresponding to GAN-RoBERTa-Small, GAN-RoBERTa-Large, GAN-DetectGPT-Small and GAN-DetectGPT-Large, respectively. A more detail variants information is provided in Table 4.1.

Table 4.1: GAN-GPTZero Variants Models

The table encompasses the structural components present in each variant. It should be highlighted that all four variants utilize GPT2-Large as their Generator G.

Full Name	Abbreviation	Perturbator	Discriminator
GAN-RoBERTa-Small	GR-Small	–	RoBERTa-Base
GAN-RoBERTa-Large	GR-Large	–	RoBERTa-Large
GAN-DetectGPT-Small	GD-Small	T5-Large	GPT2
GAN-DetectGPT-Large	GD-Large	T5-Large	GPT2-Medium

To align with the training configuration, the experimental results presented in this chapter will exclusively focus on Nucleus Sampling decoding strategy.

4.4 Performance Comparison

The assessment of our GAN-GPTZero is divided into two main components: generation evaluation and discrimination evaluation. The generation evaluation includes perplexity and distribution similarity, while the discrimination evaluation focuses on comparing the AUROC and comparing against human judgment. These aspects are detailed in the following subsections, in sequence. In addition to these statistical comparisons, Appendix H offers two illustrative examples for a more comprehensive understanding.

4.4.1 Comparison on Perplexity

A validation of the GAN-GPTZero Generator G is conducted in comparison with the original GPT2-Large model, with perplexity serving as the evaluation metric. Importantly, the BBC News dataset utilized for calculating perplexity is distinct from the training datasets for both original GPT2 and GAN-GPTZero Generator G, thereby ensuring a more unbiased assessment. Upon evaluation, it is observed that for both RoBERTa style and DetectGPT style variants, the models with smaller architectures exhibit higher perplexity scores. However, these scores remain lower than those recorded for the original GPT2 model, as delineated in Table 4.2. Our empirical analysis reveals that the GAN-GPTZero variants consistently outperform the original GPT2 model in terms of lower perplexity scores. This finding suggests that GAN-GPTZero enhances the model's generalizability across diverse topics. Specifically, the finetuned Generator G appears to possess a broader competency in topics coverage compared to the original GPT2 model.

Table 4.2: Perplexity between variants and original GPT2

With a larger architecture, the model exhibits diminished perplexity scores, indicating an augmentation in accuracy during the prediction of subsequent words in accordance with the experimental dataset.

	GR-Small	GR-Large	GD-Small	GD-Large	GPT2-Large
Perplexity	15.17	13.28	15.08	12.89	19.93

4.4.2 Comparison between Human and Model Distribution

To rigorously assess the generative capabilities of the models under consideration, a two-fold distributional analysis is employed: the empirical distribution of the dataset and an approximated distribution of the model outputs. For the empirical distribution of the dataset, the tiktoken library [52] for tokenization is employed. Each token's frequency of occurrence is computed across the entire dataset, and the resulting approximated histogram density serves as a quantitative representation of the data distribution.

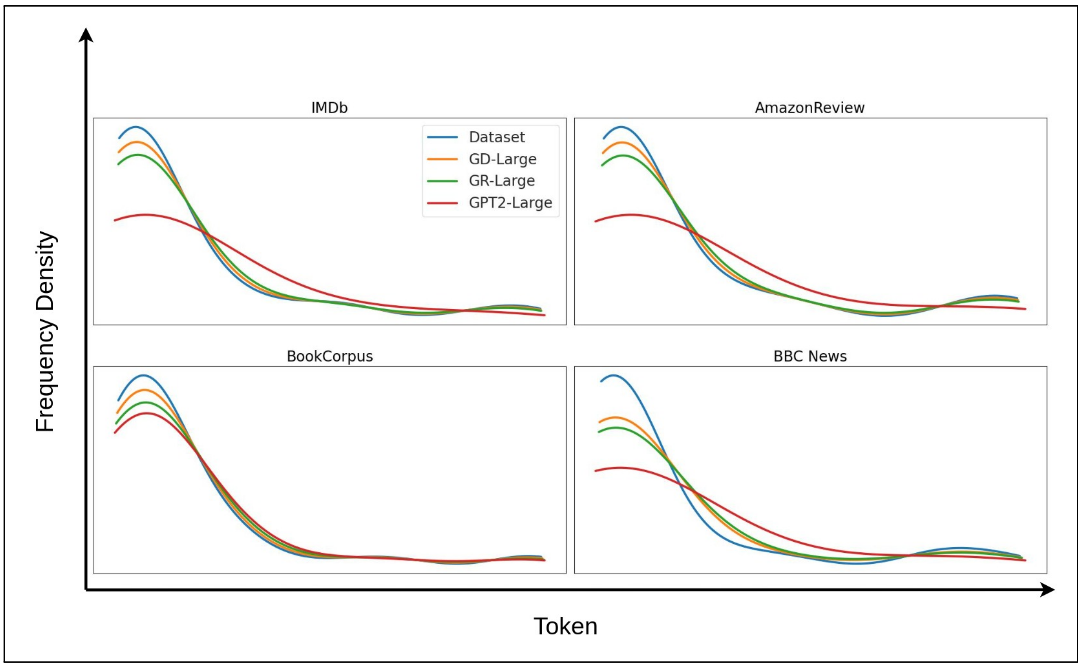


Figure 4.2: Approximation of Frequency Density per Token.
A comparative analysis of four distinct datasets: two of the GAN-GPTZero fine-tuning dataset, a pretrained dataset and a novel, previously unseen BBC News dataset. It is observed that the GAN-GPTZero variants exhibit a higher degree of distributional similarity compared to the original GPT2-Large model in both scenarios.

For the model distribution, direct estimation poses a computational challenge. In order to approximate, the generated text S^g is used to calculate the frequency according to the tokens. The resulting histogram density of which is assumed to approximate the model's underlying distribution.

Both the original GPT2-Large model and the two largest GAN-GPTZero variants are evaluated, utilizing the IMDb, AmazonReviews, BookCorpus as well as the independent, never-before-seen BBC News dataset to ensure a comprehensive and unbiased comparison. The comparative distributions are illustrated in Figure 4.2, GAN-GPTZero variants have a more similar distribution than that of the original GPT2. The outcome is consistent with expectations, given that the original GPT2 has not been trained on IMDb and AmazonReviews dataset. Intriguingly, this pattern of heightened similarity persists even when evaluate on the BookCorpus and BBC News dataset, indicating that the GAN-GPTZero variants have a more human-like 'curvature' and maintain a closer alignment with the underlying human distribution. These empirical observations underscore the efficacy of employing GAN-GPTZero as a mechanism to guide the Generator G toward a distribution that more closely approximates that of human.

4.4.3 Comparison on Accuracy: AUROC

For each experimental variant, the model performance is assessed with the AUROC. This metric serves as an indicative measure of a classifier's ability to correctly prioritize a machine generated example over a human written one when both are randomly selected. It is important to note that all experimental setups maintain an equal distribution of positive (machine generated) and negative (human written) samples.

In comparative analyses involving the baseline DetectGPT style and RoBERTa style methods on our experimental dataset, as shown in Figure 4.3, it is found that the GAN-GPTZero facilitates a more optimal equilibrium, effectively balancing both domain-generic applicability and predictive accuracy. This implies that our model contributes a nuanced architecture of adaptability, enhancing both the genericity and accuracy of text classification models.

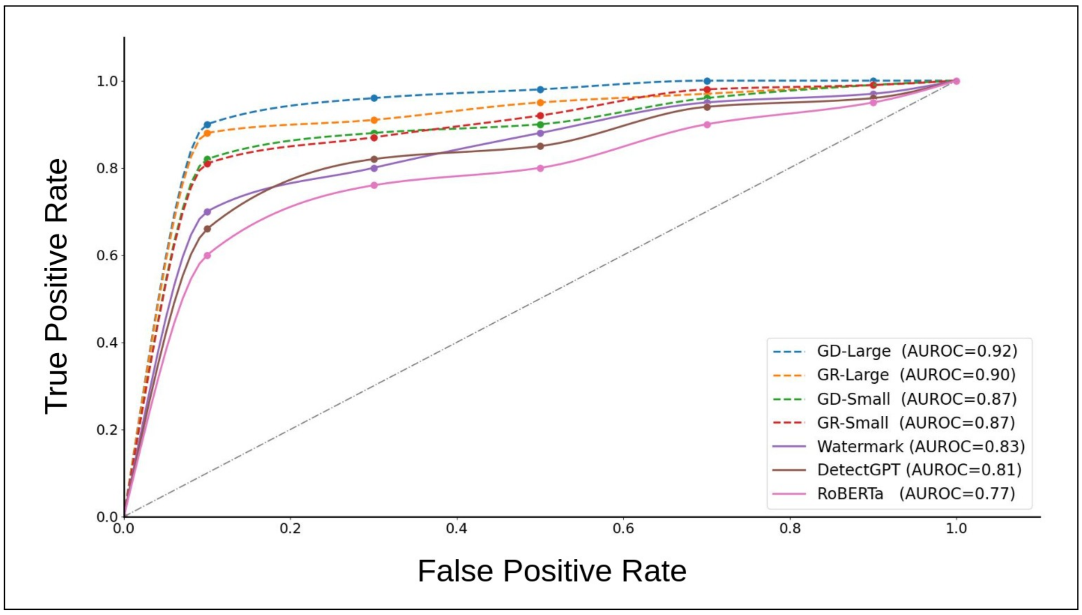


Figure 4.3: AUROC on Two Datasets
It illustrates the AUROC for text samples of length 256 across various detectors, including GAN-GPTZero variants, etc. The results indicate that GAN-GPTZero variants consistently outperform other detectors. Notably, among the GAN-GPTZero variants, GD-Large and GR-Large emerge as the most proficient in terms of detection accuracy.

As depicted in Figure 4.4, the AUROC scores of GD-Large exhibit a positive correlation with the increasing length of the input text. This observation is consistent with findings reported in existing literature, specifically [1], [44], and [50]. This suggests that while machine generated text is becoming increasingly indistinguishable from human written text, the likelihood of accurate detection escalates when the text length is sufficiently extended.

4.4.4 Comparison against Human Judge

In our study, the effectiveness of various GAN-GPTZero variants on their abilities to disseminate disinformation are assessed by utilizing our modified experimental dataset, composed of both human written and machine generated texts. Each dataset contains 2,000 textual pieces. For the evaluation, a cohort of 10 individuals, consisting of both colleagues and friends, was assembled to review

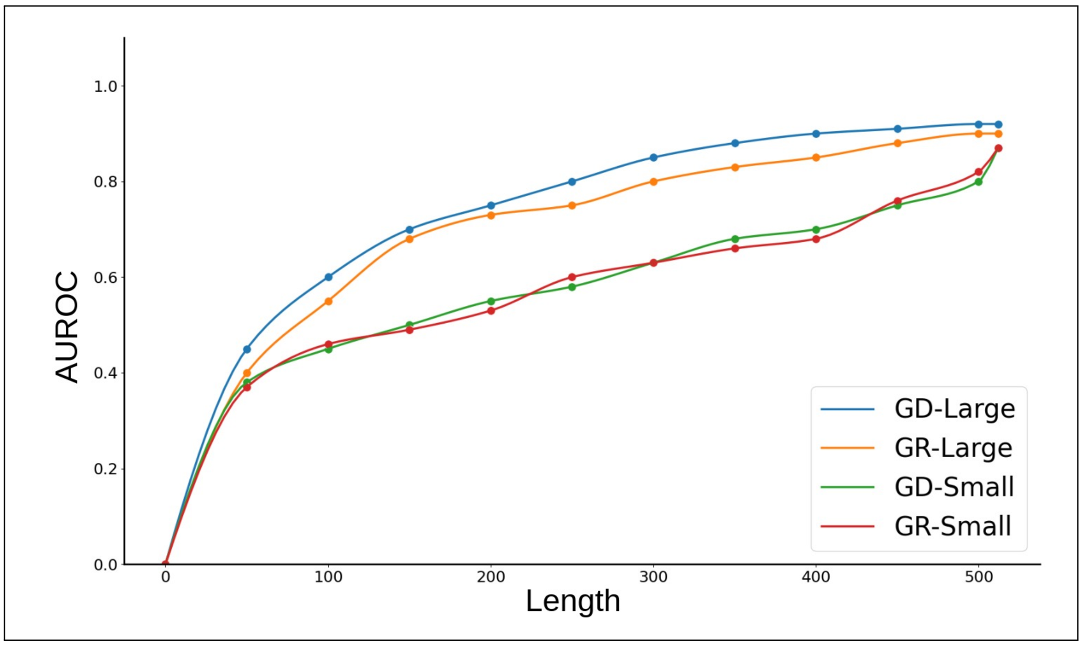


Figure 4.4: Accuracy change in accordance with Text Length
It illustrates the variation in the AUROC score for GD-Large in accordance with text length within the experimental dataset. It indicates that detection accuracy improves with increasing text length, with GD-Large exceeds the **90%** accuracy threshold at length of 400. Additionally, the figure underscores the considerable impact of training on random-length examples, particularly in enhancing detection accuracy for shorter text segments.

a random sampling of 200 texts from the experimental pool.

As depicted in Figure 4.5, the human evaluators occasionally misclassified human written articles from experimental dataset as machine generated. Interestingly, it is observed an inverse relationship between the complexity of the Generator G and the accuracy of human evaluation: as the Generator G becomes larger, human evaluators find it more challenging to distinguish the origin of the text. Conversely, the performance of GAN-GPTZero Discriminator D improves with a more complex Generator G.

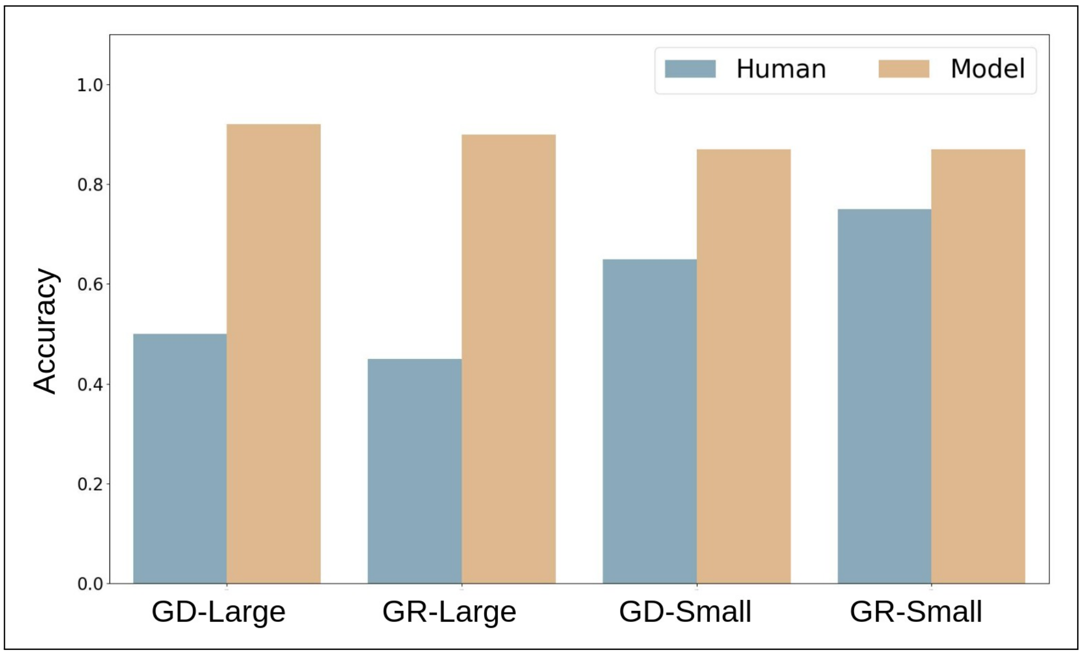


Figure 4.5: Comparison with Human Evaluation

Our findings reveal an inverse correlation between Generator G complexity and human evaluation accuracy: as the Generator G becomes larger, the capacity for human detection diminishes. Intriguingly, the GAN-GPTZero Discriminator D exhibits an opposite trend, becoming increasingly proficient in text classification as the complexity of the Generator G escalates.

4.5 Limitations of Other Detectors

4.5.1 Limitations in DetectGPT Detector and RoBERTa Detector

Classification algorithms utilized in AI generated text detection domain can be broadly categorized into two paradigms: zero-shot learning methods, exemplified by DetectGPT, and supervised learning methods, epitomized by RoBERTa. The fundamental distinction between these two paradigms resides in their domain sensitivity. Our empirical results reveal that models finetuned on generated text incorporated with the GAN architecture, demonstrate superior performance on the datasets they were trained and finetuned on. Conversely, zero-shot models manifest moderate yet interesting accuracy when tested on novel datasets. This observation suggests that each paradigm is engaged in an optimization trajectory unique to its objectives. Specifically, supervised methods are geared towards maximizing accuracy within specific, well-defined datasets, while zero-shot methods aim to achieve a balanced trade-off between generic applicability and acceptable levels of accuracy. Additionally, as hypothesized in section 2.4, the enhanced performance observed in GAN-DetectGPT variants might be attributed to the richer intrinsic ’curvature’ present. The greater the diversity and complexity of this ’curvature’, the more the generated text resembles human-like language.

4.5.2 Limitations in Watermark Detector

Watermark techniques offer a refined approach to identifying the output generated by LLMs by embedding unique patterns within the text. One such technique, soft watermark [32] categorizes tokens into ’green’ and ’red’ lists, thereby guiding the LLMs to sample tokens predominantly from the ’green list’ based on the prefix tokens. These watermark patterns are generally inconspicuous to human readers. On the other hand, DetectGPT approach generates semantically similar rephrasings that can reveal the model’s inherent biases toward specific phrasings, which could be considered as an implicit form of watermarking. Essentially, LLMs inadvertently expose their machine origins when they fail to perfectly emulate human-like text. Integrating the watermarking technique into a GAN-DetectGPT hybrid method has the potential to enhance the effectiveness of machine generated text detection.

Chapter 5

Conclusion

This chapter summarizes our key findings, particularly focusing on the comparative performance of GAN-GPTZero vis-à-vis other detection models. The discussion of emerging methodologies is also included, notably the watermarking technique, and their potential implications for the field of disinformation detection. In addition to our conclusions, the future research directions, including the applicability of detection algorithms in large language models are outlined. Furthermore, the prospects for improving the accuracy of detecting machine-generated text are discussed.

5.1 Limitations in GAN-GPTZero and Future Work

The primary limitations of the GAN-GPTZero model lie in the challenges associated with hyperparameter tuning and the selection of an appropriate loss function for training. Due to its adversarial nature, the GAN model is highly susceptible to collapse, particularly when the Discriminator D is overly powerful during the initialization phase. To mitigate this, the introduction of a dynamic weight-clipping mechanism for the Discriminator D is provided, initially suppressing weight updates and gradually annealing this constraint to zero. Furthermore, while nucleus sampling has empirically produced the most favorable outcomes, it still lacks robust theoretical justification—a point also observed by other researchers in the [1] community.

Regarding the loss function, it is found that using Binary Cross-Entropy Loss for the Discriminator D does not necessarily guide the model to converge to the ideal dataset distribution and may even induce model collapse. To be more specific, BCE Loss exhibits sensitivity towards infrequent terms, potentially inducing biases towards predominant words. Additionally, the disparity between BCE Loss and intuitive metrics such as accuracy may lead to models' overconfidence in their generated outputs. Plus, the presence of exposure bias contributes to instability during the inference phase. Consequently, models trained with BCE Loss is frequently navigated to the intricate balance between diversity and quality, as delineated in [41].

Our experiments with WGAN Loss [15] have not yielded significant improvements at present, but it is hypothesized that careful modifications could theoretically lead to better results. Future work will concentrate on refining the GAN architecture, including a systematic methodology for optimal hyperparameters selection. Another avenue of focus will be the integration of GAN-GPTZero and watermark methods to develop more effective text perturbation strategies.

5.2 Summary

This dissertation introduces a novel method, GAN-GPTZero, employing a Generative Adversarial Network framework to integrate Generator G, Discriminator D and Perturbator P. Our findings, based on comprehensive evaluations against existing detectors, reveal that GAN-GPTZero outperforms its competitors in various metrics. Specifically, the Generator G closely approximates the dataset distribution, as evidenced by our experimental analysis. While our method shows promise, there remains room for further optimization. Future research will focus on refining hyperparameter settings, exploring alternative loss functions, and integrating watermark techniques with our GAN-GPTZero.

References

[1] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature, 2023.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[9] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration, 2019.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[11] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[15] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[17] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2018.

[19] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[20] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[21] Luke Metz, Niru Maheswaranathan, C. Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves, 2020.

[22] Ali Borji. Pros and cons of gan evaluation measures, 2018.

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019.

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

[25] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[26] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[27] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[28] Flavien Prost, Nithum Thain, and Tolga Bolukbasi. Debiasing embeddings for reduced gender bias in text classification. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 69–75, Florence, Italy, August 2019. Association for Computational Linguistics.

[29] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[30] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, mar 2020.

[31] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[32] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models, 2023.

[33] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models, 2023.

[34] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[35] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense, 2023.

[36] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected?, 2023.

[37] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[40] OpenAI. Gpt-4 technical report, 2023.

[41] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

[42] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[43] Fatemehsadat Mireshghallah, Justus Mattern, Sicun Gao, Reza Shokri, and Taylor Berg-Kirkpatrick. Smaller language models are better black-box machine-generated text detectors, 2023.

[44] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news, 2020.

[45] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[46] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[47] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[48] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[49] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[50] Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. On the possibilities of ai-generated text detection, 2023.

[51] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[52] OpenAI. tiktoken: A tokenization library from openai, 2023. OpenAI's tiktoken is a Python library designed to tokenize text data, a fundamental pre-processing step in many natural language processing (NLP) tasks. The library is particularly noted for its efficiency and utility in various NLP and machine learning applications. It is part of OpenAI's ecosystem of tools for artificial intelligence research and development.

[53] Y. Polyanskiy and Y. Wu. Information theory: From coding to learning, 2022.

[54] Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation, 2019.

[55] James Surowiecki. The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations., 2004.

[56] L. Ungar, B. Mellors, V. Satopää, J. Baron, P. Tetlock, J. Ramos, and S. Swift. The good judgment project: A large scale test of different methods of combining expert predictions. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2012.

[57] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers, 2021.

[58] Chenze Shao, Yang Feng, and Xilin Chen. Greedy search with probabilistic n-gram matching for neural machine translation, 2018.

[59] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, 2017.

[60] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.

[61] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation, 2017.

Appendix A

Introduction to LSTM and GRU

Long Short-term Memory units address these issues. LSTM introduces a memory cell and three gates (input, forget, and output) to control the flow of information:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{A.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{A.2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{A.3}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{A.4}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{A.5}$$

$$h_t = o_t \times \tanh(C_t) \tag{A.6}$$

Gated Recurrent Units simplify the LSTM structure by combining the forget and input gates into a single ”update gate”. They merge the cell state and hidden state:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{A.7}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{A.8}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \times h_{t-1}, x_t] + b) \tag{A.9}$$

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t \tag{A.10}$$

In the domain of text generation, these architectures capture temporal dependencies, enabling the generation of coherent and contextually relevant sentences.

LSTMs and GRUs have become more popular due to their ability to handle long-term dependencies, with GRUS often favored for their reduced computational complexity.

Appendix B

Multi-Head Attention Mechanism

In the Multi-Head Attention mechanism, Equation 2.2 is calculated multiple times in parallel with different learned linear projections of the original Q, K and V. Specifically, for each attention head i:

$$Q_i = QW_i^Q, \quad K_i = KW_i^K, \quad V_i = VW_i^V \tag{B.1}$$

where W_i^Q, W_i^K, W_i^V are the weight matrices for each head.
The outputs from each head are then concatenated and linearly transformed to produce the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{B.2}$$

where, head_i is the output of the i^{th} attention head and W^O is the output weight matrix.

Appendix C

Feasibility of Detection

C.1 Total Variance as a Metric

In the scholarly work [36], a rigorous inequality is established for the AUROC with respect to any Discriminator D. Given a collection of independent and identically distributed samples $s = \{s_i\}_{i=1}^{n}$ originating either from human written text $\int$ or machine generated text $\mathcal{M}(s)$, the inequality is expressed as:

$$\text{AUROC} \leq \frac{1}{2} + \text{TV}(\mathcal{M}, \mathcal{H}) - \frac{\text{TV}(\mathcal{M}, \mathcal{H})^2}{2} \tag{C.1}$$

where, $\text{TV}(\mathcal{M}, \mathcal{H})$ denotes the Total Variation distance between the distributions $\mathcal{M}$ and $\mathcal{H}$, and is defined as:

$$\text{TV}(\mathcal{M}, \mathcal{H}) = 1 - \exp\left(-nI_c(\mathcal{M}, \mathcal{H}) + o(n)\right) \tag{C.2}$$

where, $I_c(\mathcal{M}, \mathcal{H})$ represents the Chernoff information between $\mathcal{M}$ and $\mathcal{H}$, and is given by:

$$I_c(\mathcal{M}, \mathcal{H}) = -\log \inf_{0 \leq \alpha \leq 1} \int m^{\alpha}(s) h^{1-\alpha}(s) ds \tag{C.3}$$

Thus [36] argues that, from their observation, as the total variation distance (Equation C.2) diminishes, the best-possible detection accuracy reaches $1/2$, representing the AUROC corresponding to a Discriminator D that randomly predicts text as machine generated or human written.

While in [50], the authors argue that while the text length increases, the rate at which total variation distance approaches 1 is exponential with the number of samples for large deviation theory [53], Equation C.2 is updated as:

$$\text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n}) = 1 - \exp\left(-nI_c(\mathcal{M}, \mathcal{H}) + o(n)\right) \tag{C.4}$$

where, $\mathcal{M}^{\otimes n} = \prod^n \mathcal{M}$ denotes n-fold tensor product distribution, as does $\mathcal{H}^{\otimes n}$. Considering Equation C.3 and Equation C.4, Equation C.1 is updated as:

$$\text{AUROC} \leq \frac{1}{2} + \text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n}) - \frac{\text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n})^2}{2} \tag{C.5}$$

Plus, owing to the dataset processing inequality, the following relationship is observed:

$$\text{TV}(\mathcal{M}^{\otimes k}, \mathcal{H}^{\otimes k}) \leq \text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n}) \tag{C.6}$$

when $k \leq n$, which naturally extrapolates to:

$$\text{TV}(\mathcal{M}^{\otimes}, \mathcal{H}^{\otimes}) \leq \text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n}) \tag{C.7}$$

This demonstrates that, notwithstanding the proximity of machine and human distributions within the sentence space, the accumulation of additional sentences can potentially amplify the total variation norm, thereby facilitating feasible detection mechanisms. The authors' conclusion is that with the increase of the number of samples, $n \to \infty$, the total variation distance $\text{TV}(\mathcal{M}^{\otimes n}, \mathcal{H}^{\otimes n})$ exponentially approaches 1, hence increasing the AUROC.

C.2 Unifying Human and Machine Evaluation

The study [54] posits a nuanced argument, asserting that human evaluations are proficient in discerning quality, albeit fall short in capturing diversity, a facet where statistical evaluations (specifically, perplexity metrics) excel, albeit at the expense of quality assessment. However, a synergistic approach that integrates both machine and human evaluations facilitates a more refined differentiation between machine generated text distributions, denoted as $\mathcal{M}$, and human written text distributions, represented by $\mathcal{H}$.

Within this framework, $L(\cdot)$ signifies the evaluation score, while $\phi_{huse}(\cdot)$ embodies the unified feature mapping function, projecting onto $\mathbb{R}^d$. Notably, the human text distribution $\mathcal{H}$ is approximated through a collaborative aggregation of diverse human judgments, a methodology inspired by seminal works [55] and [56]. The overall HUSE evaluation score is $L(\phi_{\text{huse}}(\mathcal{M}, \mathcal{H}))$.

It is imperative to acknowledge that this research, published in 2019, conveys a certain reservation regarding the complexities involved in training an optimal discriminator. This apprehension stems from the inherent limitations imposed by the immaturities of research resources. Despite this, the criterion formulated within the study holds significant merit, particularly in its capacity to balance between quality and diversity. This represents a pioneering contribution within the field, laying a solid foundation for the development of more sophisticated and effective evaluative mechanisms within the domain of Natural Language Processing.

C.3 Kullback-Leibler divergence as a Metric

[57] quantifies the difference between machine generated text and human written text. It does so by constructing a divergence curve, which encapsulates a spectrum of errors that capture phenomena present in both machine and human-generated text. The divergence curve is a trade-off representation between Type I Error and Type II Error, and the AUROC provides a scalar summary of this trade-off.

- Type I Error: Where the model assigns high probability to sequences that

do not resemble human written text.
- Type II Error: Where the model's distribution does not encompass the human distribution, leading to a lack of diverse samples.

Mathematically, the MAUVE divergence curve, $C\mathcal{M},\mathcal{H}$, is defined as:

$$\mathcal{R}_\lambda = \lambda\mathcal{M} + (1-\lambda)\mathcal{H}, \quad \lambda \in (0,1) \tag{C.8}$$

$$C(\mathcal{M},\mathcal{H}) = \{\exp\left(-c \times \text{KL}(\mathcal{M} \mid \mathcal{R}_\lambda)\right), \exp\left(-c \times \text{KL}(\mathcal{H} \mid \mathcal{R}_\lambda)\right)\} \tag{C.9}$$

where KL$(\cdot)$ represents Kullback-Leibler divergence, $\mathcal{M}$ and $\mathcal{H}$ are the machine generated and human written text distributions respectively, and $\mathcal{R}_\lambda$ is a mixture of these distributions, $c > 0$ is a hyperparameter for scaling.

Contrary to the approach adopted by HUSE, the author of MAUVE utilizes a Monte Carlo estimator using samples $s \in \mathcal{M}$ and $s \in \mathcal{H}$ to overcome the fact that ground-truth probabilities are unknown. Furthermore, it is to be emphasized that MAUVE is not just another metric, it is a comprehensive measure that provides insights into various properties of generated text, such as text length, model size, and decoding algorithms. Empirical results demonstrate that MAUVE correlates well with human judgments, making it a promising tool for evaluating open-ended text generation.

Appendix D

Basics of Detectgpt

The perturbation discrepancy is interpreted as a measure of the local curvature of the log probability function near the candidate passage. The curvature is related to the Hessian of the log probability, which gives insights into the ”curvature” of the function at a particular point. The paper suggests that the perturbation discrepancy approximates the negative trace of the Hessian of the log probability function. The Hutchinson’s trace estimator can be used to estimate the trace of a matrix:

$$\text{tr}(A) = E_z z^T A z \tag{D.1}$$

where, the matrix A is the trace which the estimator is to provide an unbiased estimate of.

Appendix E

The Watermark Detector

E.1 Watermark from the Maryland

Central to the [32] is the assumption that an autoregressive language model is trained on a vocabulary V of size $|V|$. Given a sequence of tokens as input at time step t, a language model predicts the next token in the sequence by outputting a vector of logit scores $l_t \in R^{|V|}$ with one entry for each item in the vocabulary. A random number generator is seeded with a context window of h preceding tokens, based on a pseudo-random function (PRF) $f \colon N^h \to N$. With this random seed, a subset of tokens of size $\gamma\,|V|$ are "colored" green and denoted G_t. Now, the logit scores l_t are modified so that:

$$l_t = \begin{cases} l_{tk} + \delta, & \text{if} \quad k \in G_t \\ l_{tk}, & \text{otherwise} \end{cases} \tag{E.1}$$

After modifications, these logit scores can be used for any desired sampling scheme. In the simplest case, one passes the scores through a softmax [34] layer and samples from the output distribution, resulting in a bias towards tokens from G_t.

The watermark can be described by four parameters. The "hash" used to generate the green lists f with context width h, greenlist fraction γ, and the logit bias δ. After watermarked text is generated, one can check for the watermark without having access to the LLM by re-computing the greenlist at each position and finding the set s of greenlist token positions. The statistical significance of a sequence of tokens of length T can be established by deriving the z-score:

$$z = \frac{|s| - \gamma T}{\sqrt{\gamma(1-\gamma)T}} \tag{E.2}$$

When this z-score is large (and the corresponding p-value is small), one can be confident that the text is watermarked.

However, [32] argues the vanilla watermark is vulnerable. Systematically, the vanilla watermark only chooses the final word as the hash index to generate the random sequence, which could be vulnerable because the third-party observer still could learn the green lists associated with the token at position $t-1$ by searching subsequent words at position t that are less likely to appear than expected under a non-watermarked distribution. By increasing the context width h, the three modifications are mathematically defined as:

$$f_{\text{Additive\_LeftHash}}(x) = P(s\sum_{i=1}^{h} x_i) \quad h > 1 \tag{E.3}$$

$$f_{\text{Skip\_LeftHash}}(x) = P(s_h) \tag{E.4}$$

$$f_{\text{Min\_LeftHash}}(x) = \min{}_{i\in 1,\dots,h} P(sx_i) \tag{E.5}$$

where, $f\colon \mathbb{N}^h \to \mathbb{N}$ is a function that maps a span of tokens s_i onto a pseudo-random number. Each depends on a secret salt value $s \in N$ and a standard PRF $P\colon \mathbb{N} \to \mathbb{N}$.

The author indicates that a smaller context width h provides the best robustness to machine paraphrasing. At wider context widths, *Skip* and *Min* variants remain strong under attack while *Additive* suffers. This shows that the additional strength provided by *Min* and *SelfHash* (the case when $h = 1$) can be used to run longer context widths, which in turn secure the watermark.

What's more, with a modified z-test, i.d. *WinMax* which is an alternative of formulating a detection hypothesis that can be employed optionally or in conjunction with the original test and requires no modification of the generation scheme. Given a sequence of tokens, we first score the sequence on per-token basis to find the binary vector of hits $s \in \{0,1\}^T$ to reach green list, which can be converted to a partial sum representation $p_k = \sum_{i=1}^{k} s_i$. *WinMax* searches for the continuous span of tokens that generates that highest z-score. More formally, in computes:

$$z_{\text{win\_max}} = \max_{i,j \ \mid \ i<j} \frac{(p_j - p_i) - \gamma(j-i)}{\sqrt{\gamma(1-\gamma)(j-i)}} \tag{E.6}$$

E.2 Watermark from Amherst

Considering to increase the ability of defending from the paraphrasing attack, [35] demonstrates that existing detectors are vulnerable to paraphrase attacks. For these attacks, an external paraphraser model is required, as paraphrases generated by the base LLM can still be detected. The authors trained an 11B parameter paraphrase generation model called DIPPER to execute these attacks. DIPPER has two main features:

- It can paraphrase long-form text in context. Unlike most paraphrases that focus on sentence-level data, DIPPER is trained to handle paragraph-length texts.
- It provides control over output diversity. Users can control the lexical diversity and the amount of content reordering in the paraphrases.

To defend against paraphrasing attacks, the authors propose a retrieval-based method. Given a candidate text, the algorithm searches a database of sequences previously generated by the API, looking for sequences that match the candidate text semantically. This method was found to be robust against paraphrase attacks, detecting 80% to 97% of paraphrased generations across different settings. In order to detect the watermark, z-score is used as a detection algorithm:

$$z = \frac{|s|_G - \gamma T}{\sqrt{T\gamma(1-\gamma)}} \tag{E.7}$$

where, $|s|_G$ is the number of green tokens (watermarked tokens) in the sequence, γ is the fraction of tokens that are watermarked, T is the total number of tokens in the sequence.

E.3 Watermark from Stanford

[33] employs a deterministic method, mapping a sequence of random numbers (encoded by a watermark key) to a sample from the language model. This generated text should ideally be distortion-free, preserving the original text distribution. It should also be model-agnostic, meaning it can be detected without the original language model or prompt, and robust, ensuring it remains detectable even after potential perturbations.

A pivotal component is the decoder function. This function maps an element of the watermark key and a distribution over the next token to a next token prediction. This ensures that the prediction is a sample from the distribution, rendering the watermark distortion-free. For the watermarking process, it defines a relationship between the generated text and the watermark key sequence:

$$P(\text{text}) = \int 1\text{text} = \text{generation}(\xi, \text{prompt}) d\nu(\xi) \tag{E.8}$$

where, $P(\xi)$ represents the probability of generating a particular text, 1 indicates the indicator function, generator(ξ, prompt) is the function to generate watermarked text based on the watermark key sequence ξ, and $\nu\xi$ is the distribution over the watermark key sequence.

The detection of watermarked text is achieved using a test statistic that aligns a potentially watermarked text with the watermark key sequence. The objective is to compute a p-value against the hypothesis that the text is not watermarked. This ensures that the watermark remains detectable even if the text undergoes edits or cropping. The p-value is calculated as:

$$p_b = \frac{1}{T}\sum_{t=1}^{T} 1\phi(y,\xi) < \phi_t \tag{E.9}$$

where, y is a string from vocabulary V, seed sequence is from seed space Ω, ϕ is the test statistic, $\phi_{t_{t=1}}^{T}$ is the reference distribution, p_b is the output in the range $[0,1]$.

In this algorithm, $\phi(y,\xi)$ represents the test statistic for the given text and the watermark key sequence ξ. The indicator function $1\phi(y,\xi) < \phi_t$ checks if the test statistic for the text is less than a reference value ϕ_t from the reference distribution. The p-value p_b is then computed as the fraction of times this condition is true across the entire reference distribution. The goal of the p-value method is to determine if the text y is watermarked. A low p-value indicates a high likelihood that the text is watermarked, while a high p-value suggests the opposite.

Appendix F

Decoding Strategies

F.1 Decoding Strategies in Text Generation

As discussed above, the auto-regressive language generation is based on the assumption that the probability distribution of a word sequence can be decomposed into the product of conditional next word distributions:

$$P(\omega_{1:T}|W_0) = \prod_{t=1}^{T} P(\omega_t|\omega_{1:t-1}) \tag{F.1}$$

where, W_0 being being the initial *context* word sequence. The length T of the word sequence is usually determined *on-the-fly* and corresponds to the time step $t = T$ the EOS token is generated from $P(\omega_t|\omega_{1:t-1}, W_0)$.
Overall, the various decoding techniques encompass methods such as greedy search [58], beam search [59], top-k sampling [1] [42], temperature-based sampling and among others.

F.2 Greedy Search

Greedy search is the simplest decoding method. It selects the word with the highest probability as its next word:

$$\omega_t = \arg\max_{\omega} P(\omega|\omega_{1:t-1}) \tag{F.2}$$

However, the major drawback of greedy search though is that it misses high probability words hidden behind a low probability word.

F.3 Beam Search

To alleviate this problem, beam search reduces the risk of missing hidden high probability word sequences by keeping the most likely number of beams of hypotheses at each time step and eventually choosing the hypothesis that was the overall highest probability. Beam search will always find an output sequence with higher probability than greedy search, but is not guaranteed to find the most likely output. Mathematically, beam search could be represented as:

$$P(S_{t+1}) = P(S_t) \times P(\omega_{t+1}|S_t)) \tag{F.3}$$

where, ω_{t+1} is the next word in the sequence. The algorithm then ranks all extended sequences based on $P(S_{t+1})$ and retains the number of beam sequences for the next iteration.

F.4 N-gram search

Yet the beam search still has repetitions but it is solved by the introduction of *n-gram* penalties as in [60] and [61]. The most common *n-gram* penalty makes sure that no *n-gram* appears twice by manually setting the probability of next words that could create an already seen *n-gram* to 0. The essence of the *n-gram* penalty is to keep track of all *n-gram* that have appeared in the current hypothesis. If an *n-gram* is about to be repeated, the model artificially suppresses the likelihood of the word causing the repetition.

Mathematically, given a sequence and a potential next word , the adjusted probability considering the *n-gram* penalty is:

$$P_{\text{adjusted}}(\omega|S) = \begin{cases} 0 & \text{if } n\text{-gram formed by } S \text{ and } w \text{ has appeared before} \\ P(w|S) & \text{otherwise} \end{cases} \tag{F.4}$$

This adjusted probability is then used during the beam search procedure. While this approach is effective in curbing repetitions, it's pivotal to choose an appropriate value of n to ensure that the generated sequences remain coherent and contextually appropriate.

F.5 Temperature

In its most basic form, sampling means randomly picking the next word ω_t according to its conditional probability distribution:

$$\omega_t \sim P(\omega \mid \omega_{1:\,t-1}) \tag{F.5}$$

It becomes obvious that language generation using sampling is not deterministic anymore. By making the distribution $P(\omega \mid \omega_{1:\,t-1})$ sharper (increasing the likelihood of high probability words and decreasing the likelihood of low probability words) by lowering the so-called *temperature* of the softmax [34].

The temperature, denoted as τ, is a hyperparameter that modulates the original probability distribution output from the softmax. The temperature-adjusted softmax function is expressed as:

$$P_\tau(\omega_i) = \frac{\exp\left(\frac{z_i}{\tau}\right)}{\sum_j \exp\left(\frac{z_j}{\tau}\right)} \tag{F.6}$$

where, z_i represents the logits before softmax, and $P_\tau(\omega_i)$ is the temperature-adjusted probability for word ω_i.

When $\tau = 1$, the softmax remains unchanged. As $\tau \to 0$, the adjusted softmax becomes more similar to a max function, making the model more deterministic. Conversely, as τ increases, the distribution becomes more uniform, making the model's outputs more random. Adjusting the temperature allows for a trade-off between the diversity and certainty of the generated text. Proper tuning of this hyperparameter can help strike a balance between producing diverse outputs and adhering to coherent linguistic structures.

F.6 Top-K Sampling

[42] introduced a simple, but very powerful sampling scheme, called *Top-K sampling*. In Top-K sampling, the K most likely next words are filtered and the probability mass is redistributed among only those K next words. GPT2 adopts this sampling scheme, which is one of the reasons for its success in story generation.

Top-K sampling, while straightforward in its concept, has a profound impact on the quality of generated text. Given the logits from the model, the Top-K sampling method first selects the top K logits based on their values and sets the rest to negative infinity. This ensures that the subsequent softmax operation only considers these top K logits. Mathematically, the procedure can be defined as follows:

Given the logits z, for each token i not in the top K values:

$$z_i = -\infty \tag{F.7}$$

The probabilities after Top-K sampling are then computed using the standard softmax:

$$P(\omega_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{F.8}$$

This method introduces a balance between pure randomness and determinism. By focusing on the top K probabilities, the model is encouraged to generate plausible text while still maintaining an element of randomness. However, the value of K plays a crucial role. A small K may lead to deterministic outputs, while a very large K might make the outputs too random. Thus, careful selection and tuning of K are essential to achieve desired results. Furthermore, while Top-K sampling offers improved coherence over vanilla sampling, there are other methods, such as Top-P or nucleus sampling, that have been proposed to further refine the balance between diversity and relevance.

Appendix G

Example of Training Dataset

This appendix provides a sample of the GAN-GPTZero DetectGPT style training dataset, which includes the Prompt Z, Human Text S^h, Perturbed Human Text $\tilde{S}^h$. The words in bold either represent the Prompt Z or the words that have been perturbed.

G.1 Prompt

- **After the creation of the IR8 demo tape Townsend and**

G.2 Human Text

- **After the creation of the IR8 demo tape Townsend and** Jason Newsted embarked on a new project they described as ”heavier than Strapping Young Lad.” However, when the IR8 tape leaked, Newsted’s Metallica bandmates discovered the project and forbade him from participating in any more side endeavors. Left on his own, Townsend decided to proceed with the album, naming it ”Physicist.” He brought together his Strapping Young Lad bandmates for the recording, marking the only occasion this lineup appeared on a Devin Townsend album. Released in June 2000, ”Physicist” is generally viewed as a low point in Townsend’s career, with both Townsend and the band expressing dissatisfaction with the sound mixing. Feeling that he had alienated some of his fanbase with ”Physicist,” Townsend saw an opportunity to create a more personal and sincere record. Inspired during a drive across Canada with his band, Townsend set out to make an introspective album dedicated to his homeland. He produced and recorded ”Terria,” featuring Gene Hoglan on drums, Craig McFarland on bass, and Jamie Meyer on keyboards. Citing specific inspirations for the album, Townsend released ”Terria” in November 2001.

G.3 Perturbed Human Text

- After **crafting** of the IR8 demo tape Townsend and Jason Newsted **initiated embark** on a new **fresh** project they described as ”heavier than Strapping Young Lad.” However, when the IR8 tape **got** leaked, Newsted’s Metallica bandmates discovered the project and forbade him from participating in any more side endeavors. Left on his own, Townsend decided to proceed with the album, naming it ”Physicist.” He brought together his Strapping Young Lad bandmates for the recording **session**, marking the only occasion this lineup appeared on a Devin Townsend album. Released in June 2000, ”Physicist” is generally viewed as a low point in Townsend’s career, with both Townsend and the band expressing dissatisfaction with the sound mixing. Feeling that he had alienated some of his fanbase with ”Physicist,” Townsend **discovered** an opportunity to create a more personal and sincere record. Inspired during a drive across Canada with his band, Townsend set out to make an introspective album dedicated to his homeland. He produced and recorded ”Terria,” featuring Gene Hoglan on drums, Craig McFarland on bass, and Jamie Meyer on keyboards. Citing specific inspirations for the **work**, Townsend released ”Terria” in **late** 2001.

Appendix H

Intuitive Comparisons

Here are two examples generated by both the GPT2-Large and GD-Large Generator G **(ours)**, accompanied by the detection scores from various detectors as well as the associated GD-Large Discriminator D **(ours)**. For a more succinct representation, the generated sentence s^g length is limited within 100 to 200 words. The detection results consist of a prediction and a confidence score. The prediction indicates whether the detector believes the text was generated by AI or written by a Human. The confidence score shows how certain the detector is about its prediction.

It's clear from both examples that GPT2-Large is prone to generating additional noise and exhibits a tendency for repetition. The words in bold represent the Prompt Z.

H.1 Example 1

H.1.1 GPT2-Large

The Second World War, a gruesome tableau of human strife, was the backdrop for the film's opening sequence. The film's director, David Lynch, said he was inspired by the film's title, which refers to the German word for 'death' and the French word for 'death'. 'I was thinking about the German word for death,

’Kriegsmarine’, and the French word for death, ’Dieu’, and I thought, ’What if we had a movie about the war that was about the war and the people who were involved in it?” he said. ’I thought, ’What if we had a movie about the war and the people who were involved in it?’ And I thought, ’What if we had a movie about the war and the people who were involved in it?’ And I thought, ’What if we had a movie about the war and the people who were involved in it?’ And I thought, ’What if we had a movie about the war and the people who were involved in it?’ And I thought, ’What if we had a movie about the war and the people who were involved in it?’ And I thought, ’What if we had a movie about the war and the people who were involved in it?’

- DetectGPT Detector: Prediction: **AI**, Confidence Score: **98%**
- RoBERTa Detector: Prediction: **AI**, Confidence Score: **100%**
- GD-Large Detector: Prediction: **AI**, Confidence Score: **100%**

H.1.2 GD-Large Generator

The Second World War, a gruesome tableau of human strife, stands as a somber testament to the paradoxical nature of human progress. It was an era where technological advancements soared to unprecedented heights, yet those very innovations were weaponized to inflict widespread devastation. Soldiers, mere pawns in a grand geopolitical chessboard, were torn from the fabric of their everyday lives to face horrors unimaginable. Fields that once bore fruits now became graveyards of metal and flesh. Skies, once a canvas of celestial beauty, were darkened by the ominous shadows of fighter planes and bombers. Yet, amid the chaos and destruction, emerged stories of unparalleled heroism and sacrifice. Ordinary individuals, propelled by extraordinary circumstances, displayed valor that transcended borders and ideologies. Nurses, doctors, and civilians braved the storm of bullets and bombs to provide a glimmer of humanity in an otherwise inhumane setting.

- DetectGPT Detector: Prediction: **Human**, Confidence Score: **50%**

- RoBERTa Detector: Prediction: **Human**, Confidence Score: **99%**
- GD-Large Detector: Prediction: **AI**, Confidence Score: **73%**

H.2 Example 2

H.2.1 GPT2-Large

Mount Vesuvius is a stratovolcano located on the Gulf of Naples in Italy. It is the largest of the Vesuvius volcanoes, and the most active. It is the most famous of the volcanic complexes, and is the most visited by tourists. The volcano is located in the middle of the island of Sicily, and is surrounded by the island of Sardinia. The volcano is a very active volcano, and is the most active volcano in the world. It is the most active volcano in the world, and is the most active volcano in the world. The volcano is located in the middle of the island of Sicily, and is surrounded by the island of Sardinia. The volcano is a very active volcano, and is the most active volcano in the world. It is the most active volcano in the world, and is the most active volcano in the world. It is the most active volcano in the world, and is the most active volcano in the world. It is the most active volcano in the world, and is the most active volcano in the world. It is the most active volcano in the world, and is the most active volcano in the world.

- DetectGPT Detector: Prediction: **AI**, Confidence Score: **100%**
- RoBERTa Detector: Prediction: **AI**, Confidence Score: **100%**
- GD-Large Detector: Prediction: **AI**, Confidence Score: **100%**

H.2.2 GD-Large Generator

Mount Vesuvius is a stratovolcano located on the Gulf of Naples, a mere stone's throw away from the bustling city. It is both a fascinating and terrifying spectacle, drawing millions of tourists while simultaneously looming over the lives of the local population. This geological titan has shaped not only the landscape but also human history. Its most infamous eruption in 79 AD entombed the Roman cities of Pompeii and Herculaneum, leaving behind a haunting, yet informative, archaeological site. The ash-frozen tableau of daily Roman life provides an invaluable window into an ancient civilization, as if frozen in time. Yet, the volcano is far from dormant. It serves as a constant reminder of the Earth's untamed power, with scientists and volcanologists keeping a vigilant eye on its seismic activities. The rich volcanic soil nurtures the surrounding farmlands, producing some of Italy's most renowned produce like the San Marzano tomatoes. However, this bounty comes with a price; the very fertility of the land is a result of past eruptions, each layer of soil a chapter in a turbulent geological history.

- DetectGPT Detector: Prediction: **Human**, Confidence Score: **20%**
- RoBERTa Detector: Prediction: **AI**, Confidence Score: **45%**
- GD-Large Detector: Prediction: **AI**, Confidence Score: **86%**